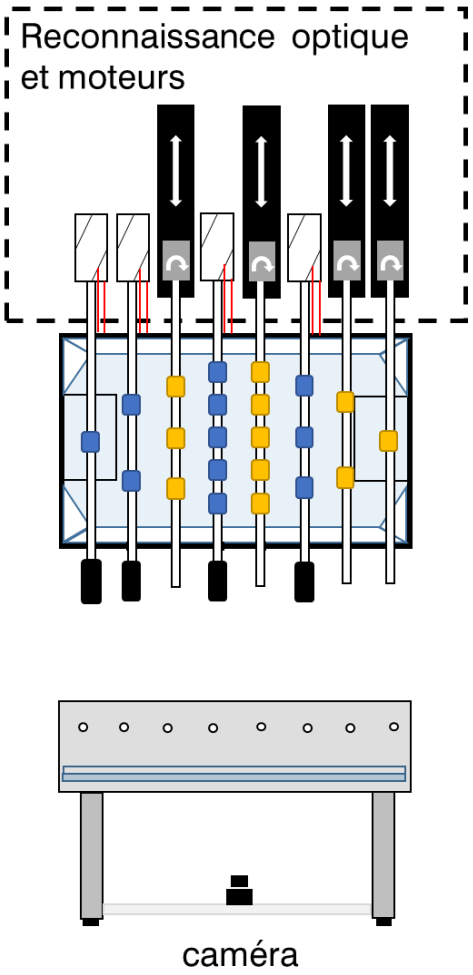
MORISSET Beñat, GRAF Rafael

Projet d'ingénierie simultanée
Babyfoot Strategy

Supervision : Dr. SALZMANN Christophe
Assistant : KAMMER Christoph



Semestre de Printemps 2016

Table des matières

Introduction

Ce projet d'ingénierie simultanée s'inscrit à la suite d'autres travaux d'ingénierie simultanée, ayant pour but final de produire un babyfoot automatisé avec une certaine intelligence artificielle.

Dans cette partie d'introduction, nous présenterons dans un premier temps en détail les dimensions du système. La complexité du logiciel augmentant chaque année, nous avons jugé bon de présenter le fonctionnement du logiciel. Puis nous passerons sur les projets des années précédentes, pour pouvoir analyser ce qui a déjà été réalisé.

Les prochains chapitres présenteront notre travail de ce semestre : La sécurité en rotation, le filtre de Kalman, les tests pour l'arrêt de la balle par blocage, un vi de tir direct avec toute la modélisation, et les sources d'imprécisions.

Présentation de la table automatisée

Mensurations de la table et des joueurs

La table de babyfoot est une « evolution G-500 » de la marque « Garlando ». La surface jouable (intérieure) est de 702 fois 1'215 mm sur un terrain transparent en plexiglas. Pour permettre à la balle de revenir sur le terrain, le bord est agrémenté d'un rebord de plastique blanc, ce qui laisse la caméra « aveuglé » lorsque la balle se retrouve sur ce dernier.

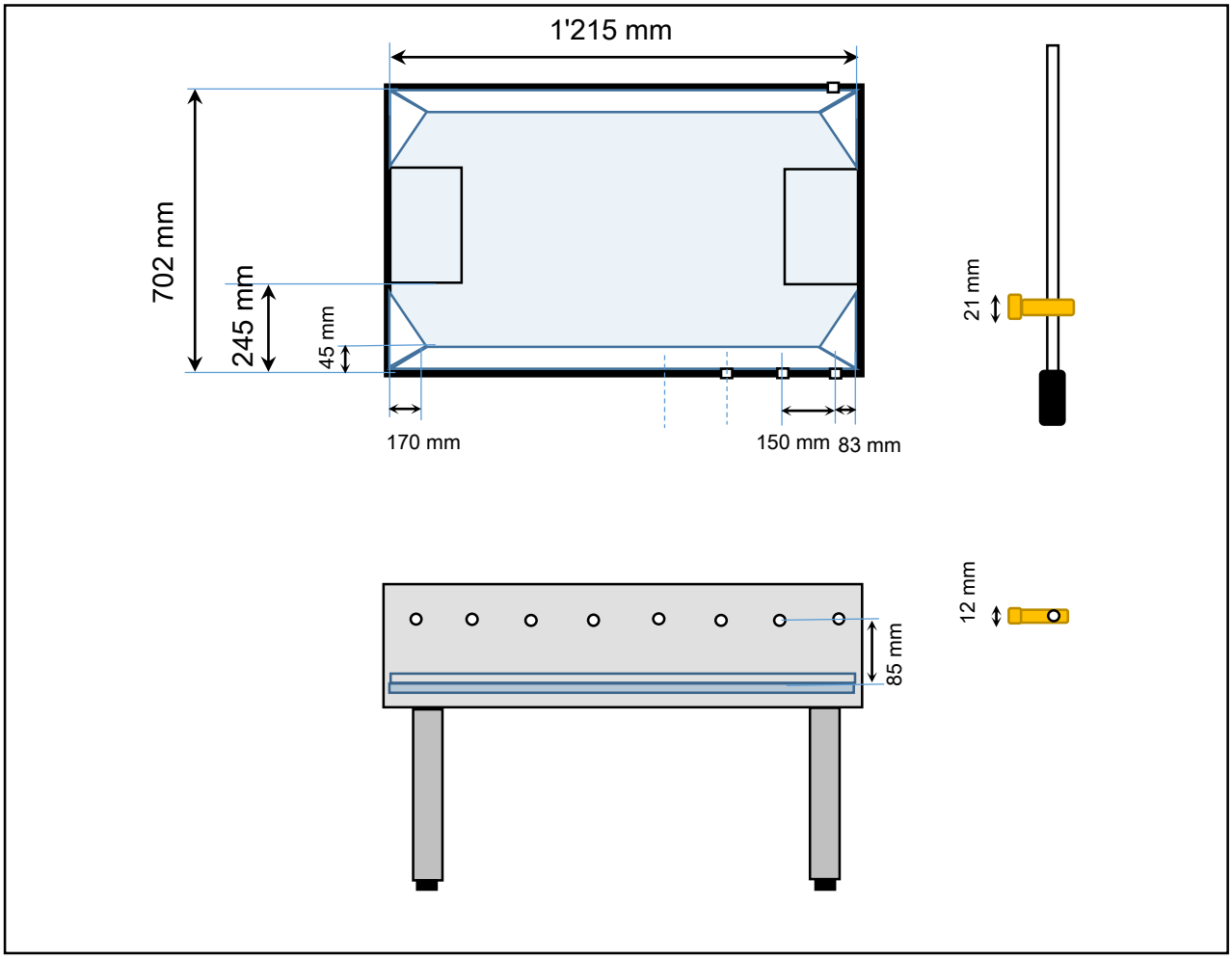


Figure 1

Les mesures principales sont présentées dans la Figure 1. Pour une meilleure lisibilité, les moteurs du côté supérieur (ou se trouve la côte 1'215 mm) ne sont pas représentés tout comme la caméra entre les 2 pieds. En annexe 1, se trouvent encore d'autres valeurs (déplacement maximal et rayon de la balle).

Logiciel et algorithme

Le logiciel utilisé principalement, et qui fait tourner le babyfoot est LabVIEW 2014. Le VI principal se nomme Global_main. Nous allons essayer de présenter succinctement, mais pas complètement son fonctionnement.

Nous appliquerons le code suivant :

- VI
  - Structures
    - VI in structure
      - Structure in structure
    - VI in structure
      - VI in VI
      - Structure in VI

- Global_main
  - Stacked sequence [0], initialisation
    - Open the camera
      - Flat sequence (0)
        - "Perform calibration ?" (l'utilisateur choisit s'il veut calibrer, via une fenêtre pop-up)
          - Machine state for camera calibration [true], (false ne fait rien)
            - While loop (attend le OK de l'utilisateur)
              - Case structure : color calibration or exposure calibration
      - Flat sequence (1)
        - "Perform calibration?"
          - Machine state for position calibration, (une succession de vi et de séquences qui font une mise à zéro de la position en Y des joueurs, et qui demande à l'utilisateur de mettre les joueurs dans une position verticale)

  - Stacked sequence [1]
    - Vision loop [2 ms, mais attends que la balle est trouvée]
      - Flat sequence qui affiche le nombre de FPS (frames per second)
      - Case structure [track object]
        - Find best score : prend la meilleure corrélation de l'algorithme de tracking de la balle si il trouve 2 balles
      - Case structure [true], si la balle est trouvée, on cherche son centre, (on convertit en unités réelles [mm]), et on corrige diffraction et l'effet de distorsion de la lentille
    - Display loop [20 ms], affiche les 10000 derniers points pris par la boucle
    - Control loop [1 ms]
      - Get_opponents_measurments, filtre les données pour pouvoir ensuite afficher les positions des opposants
        - Case [true:manual operation, false :automatic, uses strategy]
          - Basic_Strategy, ce VI, choisit comment les joueurs motorisés doivent réagir. Il choisit lequel des joueur s'occupe de la balle, et de la stratégie qu'il doit appliquer (se mettre à 90° si la balle est derrière, shooter si elle est assez proche …)
      - Bar_control_iteration, permet de contrôler les moteurs rotationnels
        - Control_IOs_update
          - Formula node, prend les paramètres du PID pour le contrôle
          - Case structure, permet la mise à zéro des moteurs linéaires manuellement

L'architecture des VI, lors du début du semestre de Printemps 2015, à l'exception de ceux issus de la librairie LabVIEW sont illustrés ci-dessous dans la Figure 2.

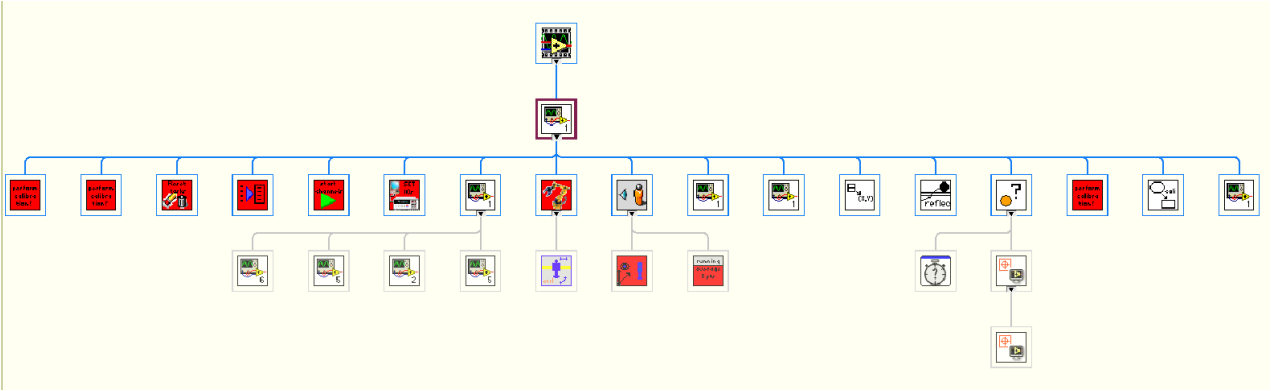


Figure 2

Projets des années précédentes

« Partie vision de la balle » (JORDAN benoît, ZILLIOX Thomas, printemps 2014)

Dans ce projet les étudiants ce sont concentrés sur le choix de la caméra à utiliser puis à l'algorithme de détection de la balle. Pour avoir une vitesse d'acquisition la plus grande possible, leur choix s'est porté sur la caméra de modèle MQ003CG – CM de la marque Ximea (200 fps mais au détriment de la qualité de l'image (648 x 488 pixels)). De plus ils ont pallié au problème de la distorsion (type « fisheye ») des lentilles et la diffraction de la lumière sur la plaque de plexiglas, avec des VI's de la librairie LabVIEW ou produits par eux-mêmes. Pour trouver la balle, un algorithme de « tracking » a été utilisé avec un temps d'exécution de 4-6 ms. Malheureusement lorsqu'il perd la balle des « yeux », l'algorithme cherche dans le proche voisinage et risque de ne pas retrouver la balle, pour résoudre ce problème récurrent, ils ont implémenté un VI de LabVIEW de « pattern matching » qui trouverait un motif en moins de 30 ms, mais sur toute l'image. La boucle principale aurait un temps moyen d'exécution de 5-6 ms.

Toutes les valeurs des distances de l'algorithme étaient en pixels et la conversion pixel → mm se faisait en multipliant la distance en pixels par un facteur 1.84.

Dans leur paragraphe 5.6, Problèmes et prochains travaux, ils ont fait les analyses suivantes :

- Le filtre numérique est très dépendant de la luminosité ambiante, un algorithme se réglant automatiquement serait bienvenu, éventuellement aussi un système d'éclairage constant.
- Certains problèmes sont apparus dans la correction de la distorsion, une focale fixe ou des mesures plus précises seraient envisageables.
- La précision sur la position de la balle ~0.5 mm pourrait être problématique.
- Une discrétisation de la surface de jeu pourrait augmenter vitesse ou précision de l'algorithme.

« Fast Positionning » (MÉTRAILLER Simon, automne 2014)

Les points abordés par ce rapport :

- insertion des moteurs linéaire (à la place d'un moteur rotationnel précédemment installé).
- le dimensionnement des pièces mécaniques, et de la table sur laquelle les moteurs sont installés.
- la communication entre le contrôleur et les moteurs.

- le tir avec un angle, d'objets situés de l'autre côté du babyfoot avec une balle statique initialement. La précision de la démarche a été résumée dans une table.

« Babyfoot Stratégie » (DÉCOSTERD Emilie, MOTTIER Valentin, printemps 2015)

Ce rapport s'occupe de mettre ensembles les parties vision et stratégie dans le programme LabVIEW. Le principe de fonctionnement de la lecture des données et de l'écriture y est décrit, tout comme la calibration des contrôleurs. La création d'un filtre ARW a également été ajouté (par contre le programme actuel n'est plus exactement le leur). C'est également dans ce rapport que les coefficients de PID/PD ont été sélectionnés (pour les moteurs rotationnels). La position désirée serait atteinte en 50 ms.

Leurs propositions d'améliorations sont les suivantes :

- Retravailler la plaque PCB
- Créer un positionnement automatique vertical des joueurs
- Améliorer la conversion de tension, et sa fiabilité
- Mettre en commun les différentes parties du VI

« Vision des adversaires » (DREYFUS Paul-Arthur, PHAM-BA Son; printemps 2015)

Dans ce rapport, ils décrivent leur démarche dans la conception permettant de connaître la position des joueurs adverses (humain).

Dans le système choisi, le bruit ne dépasserait pas les 0.25 mm, pour un temps de réponse inférieur à 0.9 ms.

« Partie Stratégie » (BLANC Pierre-Benoît, JALON Gildas; printemps 2015)

Ce projet de nature théorique, propose des algorithmes et des VI de stratégie. Dès que la position et la vitesse de la balle seront connues de manière précise, ce rapport très complet pourra être utile.

Sécurité en rotation:

Introduction:

Il arrive parfois que l'ordinateur coince la balle sous un joueur en voulant tirer. A cause du PID et son terme intégrateur, le programme commande aux moteurs de pousser vers la balle avec la puissance maximale. Suite à ce constat, qui pourrait endommager les systèmes mécaniques (accouplements, réducteurs ou moteurs), nous avons développé un algorithme qui détecte ce genre de blocages et annule les commandes du moteur le cas échéant.

Fonctionnement:

Le programme détecte si l'erreur entre la position angulaire voulue et la position angulaire actuelle est grande (> à 10°), si c'est le cas pendant plus de 35ms (plus de 3 points supérieurs à 10, Figure 3), la sécurité s'active et met la commande à 0, et une led rouge s'allume dans le Front Panel (image à droite du texte). Il est toujours possible de réactiver la commande en bougeant les joueurs dans la position que le joueur n'a pas pu atteindre.

Maintenant reste le cas lorsque l'erreur est inférieure à 10°. Cette dernière limite a été choisie en sortant un fichier texte avec les valeurs de l'erreur toutes les 10ms, et en observant une légère suspension dans les alentours de 7° à 550ms (Figure 3). De plus, si l'usage d'un VI d'arrêt de la balle via un léger blocage serait réalisé, nous éviterions d'éteindre la commande pour cette stratégie.

Pour des valeurs inférieures à 10°, nous avons donc choisi de diminuer la force appliquée, de mettre la commande à 0.25 (choix expérimental).

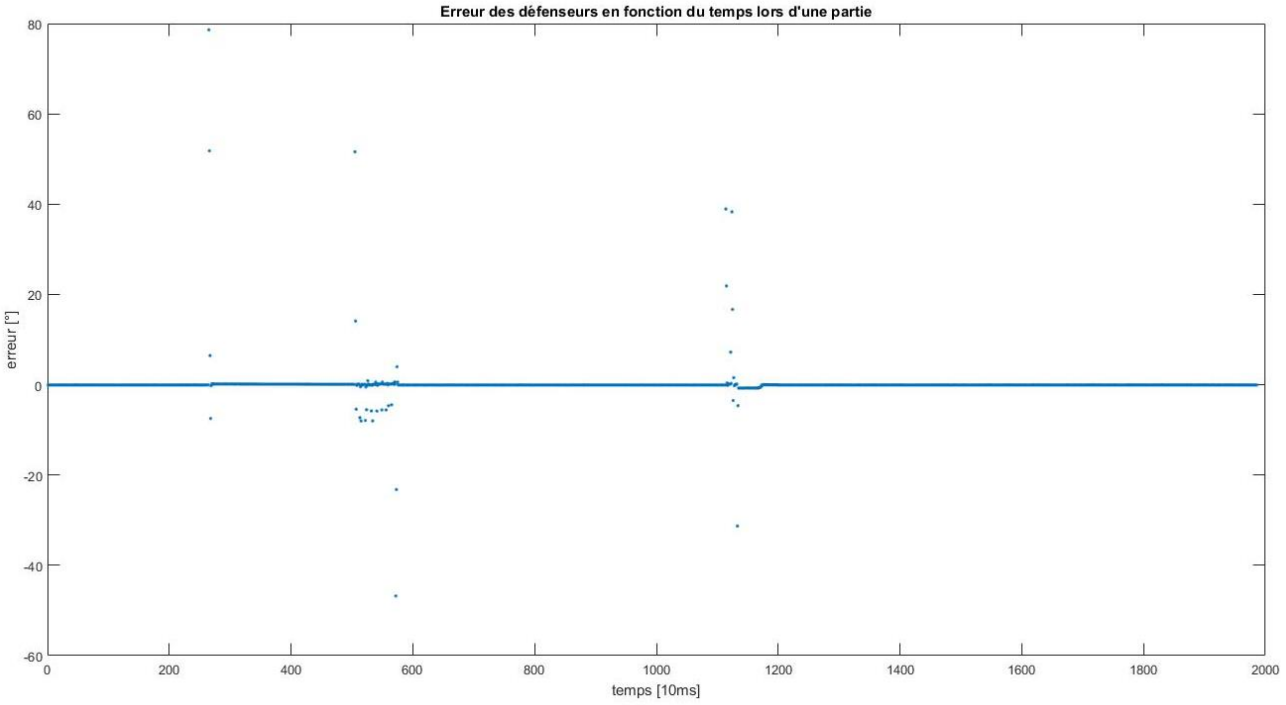


Figure 3

Filtre de Kalman

Introduction et théorie[1]

Suite au constat de l'imprécision des valeurs de la position et de la vitesse de la balle (méthode d'Euler avec moyenne sur les 5 dernières valeurs), il a été décidé d'utiliser un filtre de Kalman. Ce dernier permet, à partir de données bruitées et d'un modèle mathématique, de filtrer la position et de sortir la vitesse de la balle.

Le Filtre fonctionne en actualisant deux variables à chaque itération : L'estimation de l'état le vecteur $\overline{x_k}$ (comportant les coordonnées et la vitesse de la balle), et la matrice de covariance de l'erreur $\overline{\overline{P_k}}$.

Le filtre est composé en deux étapes, la première est la prédiction (d'où l'indice p) :

$$\overline{x_{p,k}} = \overline{\overline{A_k}}\,\overline{x_{k-1}} + \overline{\overline{B_k}}\,\overline{u_{k-1}}$$

$$\overline{\overline{P_{p,k}}} = \overline{\overline{A_k P_{k-1}}}\,\overline{\overline{A_k^T}} + \overline{\overline{Q_k}}$$

Avec :

- $\overline{\overline{A_k}}$: la matrice qui relie l'état précédent à l'actuel k
- $\overline{\overline{B_k}}$: la matrice qui relie l'entrée de la commande u à x
- $\overline{u_k}$: le vecteur de l'entrée de commande, (ou comment une perturbation est appliquée dans le système)
- $\overline{\overline{Q_k}}$: la matrice de covariance du bruit de process

Le but de cette étape est de calculer la valeur prédite par le modèle, elle ne prend pas en compte des mesures.

La deuxième étape, la mise à jour, permet d'estimer le vecteur $\overline{x_k}$, et la matrice de covariance de l'erreur $\overline{\overline{P_k}}$; en interpolant entre la prédiction et les valeurs mesurées, via une matrice K, le gain de Kalman.

$$\overline{x_k} = \overline{x_{p,k}} + \overline{\overline{K_k}}\,(\overline{y_k} - \overline{\overline{H}}\,\overline{x_{p,k}})$$

$$\overline{\overline{P_k}} = (\overline{\overline{I}} - \overline{\overline{K_k}}\,\overline{\overline{H}})\,\overline{\overline{P_{p,k}}}$$

Avec :

- $\overline{y_k}$: l'observation de la position et de la vitesse à l'instant donné
- $\overline{\overline{H}}$: matrice sortant les paramètres de $x_{p,k}$voulus
- $\overline{\overline{K_k}}$: gain de Kalman $= P_{p,k}H_k^T S_k^{-1}$, où $S_k = HP_{p,k}H^T + R_k$
- $\overline{\overline{R_k}}$: matrice d'estimation de la covariance de l'erreur

Les équations précédentes sont intuitives sur le fonctionnement du filtre, suivant la valeur de la matrice K, on va plus se fier à la prédiction ($\overline{\overline{K}} \to \overline{\overline{0}}, \overline{\overline{R}} \gg \overline{\overline{P}}$), ou alors vers les valeurs observées ($\overline{\overline{K}} \to \overline{\overline{I}}, \overline{\overline{R}} \ll \overline{\overline{P}}$).

[1] Deux principales sources : https://en.wikipedia.org/wiki/Kalman_filter, consulté le 09.05.2016, précis mais complexe. Et http://www.ilectureonline.com/lectures/subject/SPECIAL%20TOPICS/26, consulté le 02.05.2016, simple mais rempli d'erreurs.

Filtre implémenté

Dans notre cas, nous avons simplifié notre modèle. Le vecteur u_k et la matrice B_k étant difficiles à modéliser - à cause de la relation complexe entre joueurs et balle dû a la géométrie du joueur si la balle n'est pas sous l'axe de ces derniers - nous les avons écartés (ce qui va créer des problèmes ultérieurement). De même, nous avons retiré la matrice Q_k (qui reviendra sous la forme de P_{add}). Étant intéressés par la position et la vitesse, la matrice H équivaut à la matrice identité I. Ce qui nous donne les équations suivantes :

Prédiction :

$$\overline{x_{p,k}} = \overline{\overline{A_k}}\,\overline{x_{k-1}}$$

$$\overline{\overline{P_{p,k}}} = \overline{\overline{A_k}}\,\overline{\overline{P_{k-1}}}\,\overline{\overline{A_k^T}}$$

Mise à jour :

$$\overline{x_k} = \overline{x_{p,k}} + \overline{\overline{K_k}}\,(\overline{y_k} - \overline{x_{p,k}})$$

$$\overline{\overline{P_k}} = (\overline{\overline{I}} - \overline{\overline{K_k}})\,\overline{\overline{P_{p,k}}}$$

$$\cdot\,\overline{\overline{K_k}} = P_{p,k}S_k^{-1} \quad \cdot S_k = P_{p,k} + R_k$$

Utilisant ce modèle et les matrices suivantes :

$$A = \begin{bmatrix} 1 & 0 & \Delta t & 0 \\ 0 & 1 & 0 & \Delta t \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \Delta t = \frac{1}{100}, R = \begin{bmatrix} 2 & 0 & 0 & 0 \\ 0 & 2 & 0 & 0 \\ 0 & 0 & 5 & 0 \\ 0 & 0 & 0 & 5 \end{bmatrix}, P_0 = \begin{bmatrix} 2 & 0 & 0 & 0 \\ 0 & 2 & 0 & 0 \\ 0 & 0 & 3 & 0 \\ 0 & 0 & 0 & 3 \end{bmatrix}$$

Nous avons pu observer une nette amélioration pour la vitesse dans une plage linéaire (Figure 4):

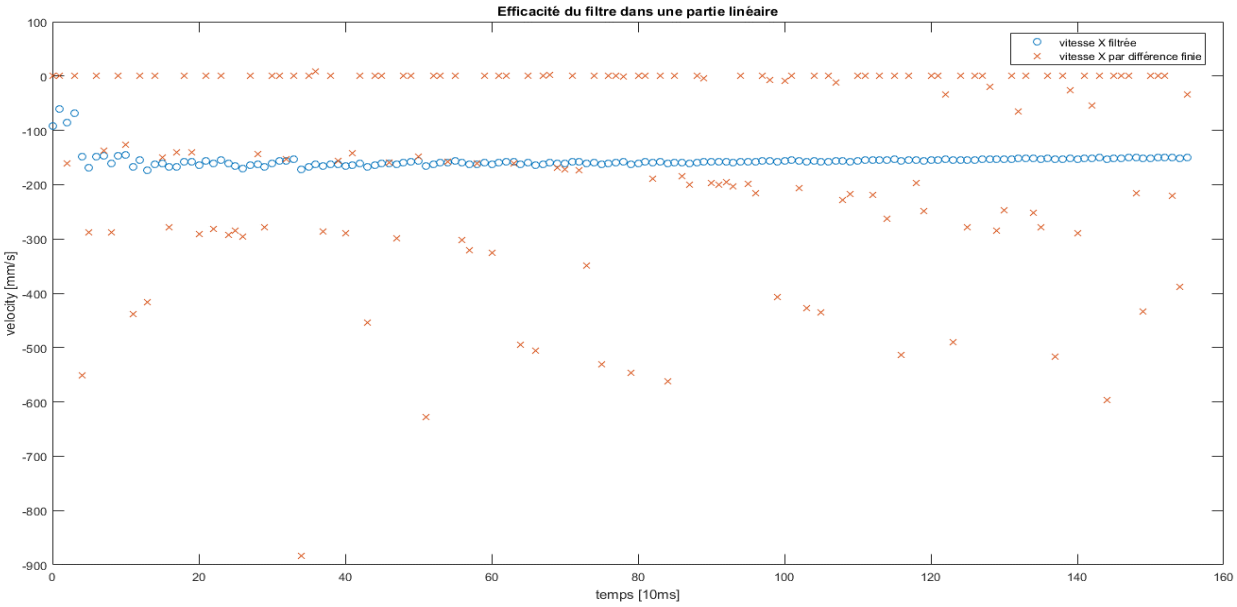


Figure 4

Le filtre a été testé avec Matlab, avec un fichier texte créé par LabVIEW, avec les coordonnées de la balle prises toutes les 10ms.

Les difficultés commencent lorsque des changements de trajectoires ont lieu. Dans la Figure 5, deux changements de trajectoires ont lieu, le premier (forme plus abrupte) est dû aux bords du système qui n'est pas couvert par la caméra, le second (de forme courbe), est dû aux contacts avec un joueur adverse. Le filtre a une certaine « inertie », qu'il faut vaincre.

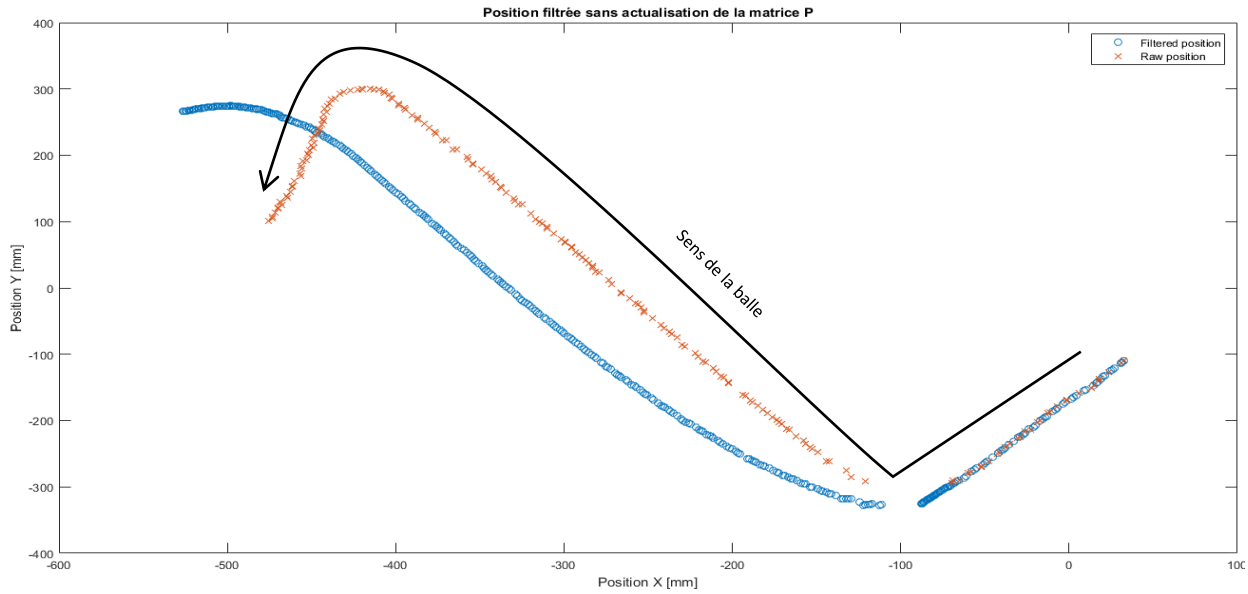


Figure 5 - en haut

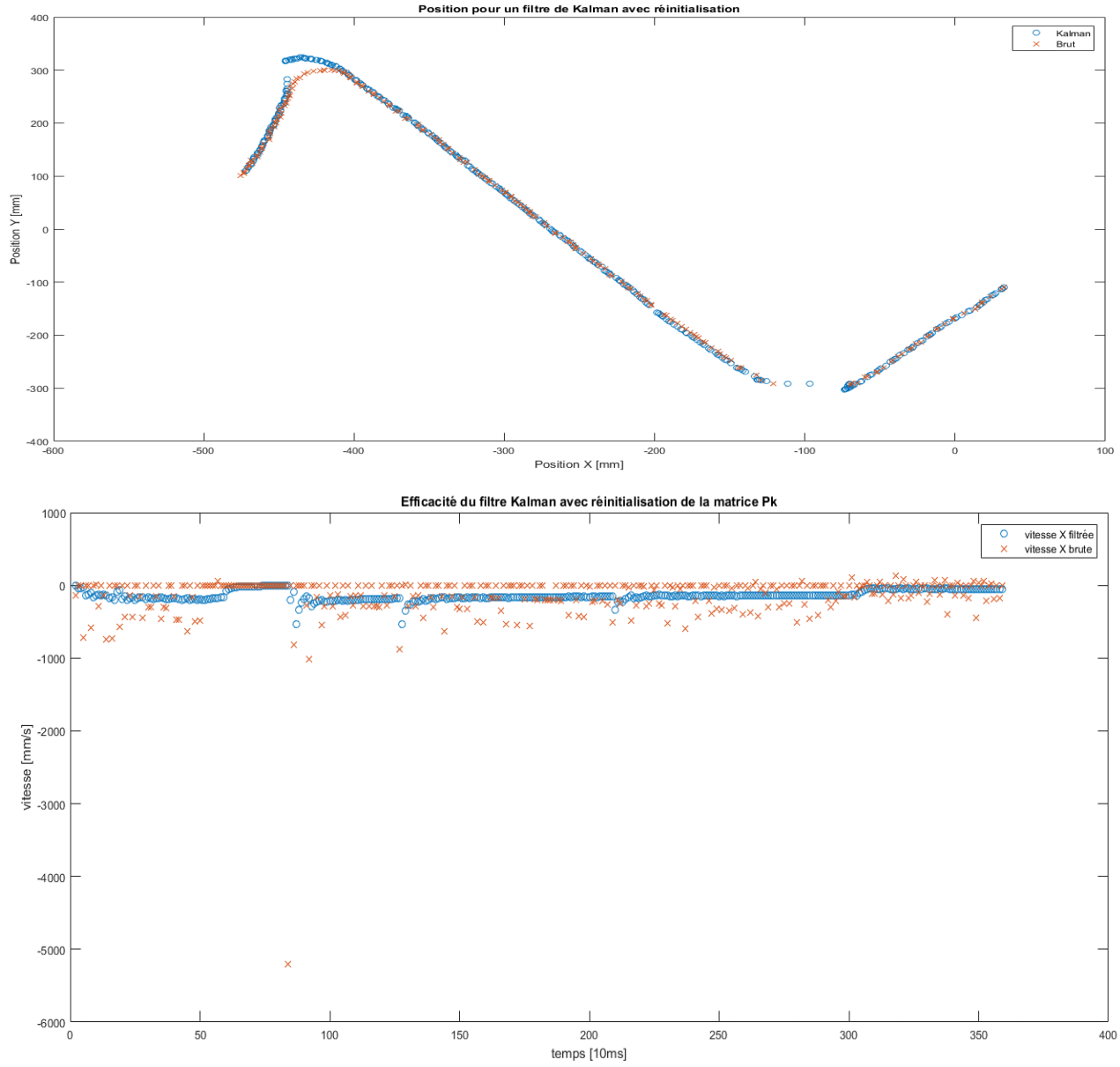


Figure 6 – positions et vitesses de la balle pour une réinitialisation de la matrice Pk

Plusieurs tests ont été effectués dans cette approche, toujours avec le programme Matlab et des coordonnées en fichier texte.

Nous avons décidé d'ajouter d'une condition. Si la distance entre le filtre excède 6mm (paramètre changé par la suite), nous faisons une réinitialisation : $\overline{\overline{P_k}} = \overline{\overline{P_k}} + \overline{\overline{P_{add}}}$. Nous avons donc testé le filtre avec l'ajout de ces conditions, et cette astuce nous permet au détriment d'un temps de convergence, de se débarrasser de « l'inertie » de notre filtre (Figure 6).

À l'aide de la Figure 6, nous pouvons également remarquer plusieurs aspects intéressants :

- La vitesse brute équivaut fréquemment à zéro. L'explication vient du fait que la position étant actualisée à une fréquence souvent inférieure à la fréquence du filtre, il y a production de doublons dans la position, et donc une vitesse nulle via différence finies.
- La vitesse brute descend une fois jusqu'à -5000 mm/s. Cette donnée vient du fait que lorsque la balle est perdue par l'algorithme, (dans ce cas sorti du champ de vision), il rend l'ancienne position pour chaque itération. Puis au moment où il retrouve la balle, il y aura une grande différence entre les positions, et si le pas de temps pour calculer la vitesse reste le même, la valeur de la vitesse monte beaucoup trop haut.

Pour répondre à ces imprécisions, nous avons choisi de calculer la position de la vitesse, de façon à ce que les doublons soient ignorés. Un simple test avec la position antérieure, avec une actualisation de la différence de temps seulement lors d'une nouvelle position suffit. Cela permet également de nous débarrasser des sauts de vitesse lorsque l'algorithme perd la balle.

L'actualisation de la position en présence de doublons n'étant pas idéale, nous avons de plus choisi d'utiliser le filtre que dans le cas où la position change. Dans le cas contraire nous faisons seulement la phase de prédiction (cela crée des plateaux pour la vitesse visible dans la Figure 7). Nous arrivons, en extrapolant de faibles vitesses, à nous positionner dans la direction de la balle avec au temps final contact entre joueur et balle (Figure 7), la balle n'étant plus toute ronde elle prenait une trajectoire curviligne.

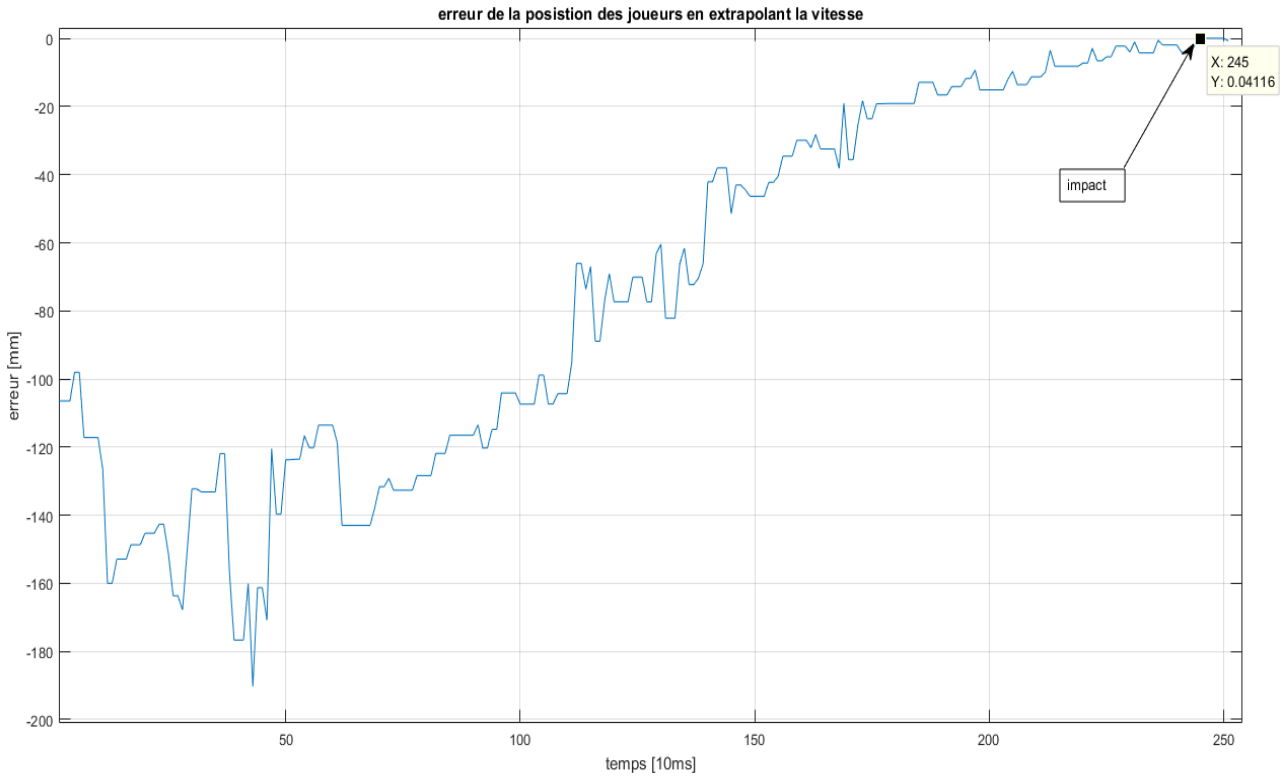


Figure 7

Dans les vitesses moyennes et rapides (>500 mm/s), l'algorithme de «tracking» perd la balle, et l'algorithme de « pattern matching » ne tourne plus qu'à 30 Hz selon le rapport de vision de la balle. Et vu le temps de réaction du filtre, une différence finie a été appliquée pour ce cas.

Récapitulation de l'algorithme ajouté :

- Global Main
  - …
  - Filter loop [1kHz], (à l'aide d'un bouton, nous pouvons encore choisir d'activer le filtre)
    - VI qui vérifie si la distance entre les dernières 2 positions, (sans compter les doublons) est plus petite que 50mm (arbitraire).
      - Case structure [distance entre 2 actualisation de la balle>50 mm, true], le filtre met trop de temps pour se réinitialiser, donc nous faisons une différence finie pour la vitesse.
        - VI speed dumb, donne la vitesse sans les doublons
      - Case structure [distance>50 mm, false], nous pouvons appliquer le filtre.
        - VI Filtre de Kalman3,
          - Case structure [2 dernières position égales=true], seulement faire la phase de prédiction, à moins que l'on a perdu la balle pour plus de 200 ms.
          - Case structure [2 dernières position égales=false], nous pouvons appliquer le filtre décrit ci-dessus.
            - Case structure [diff. entre positions brutes et filtre > 100mm] : $\overline{\overline{P_k}} = \overline{\overline{P_k}} + \overline{\overline{P_{add}}}$.
            - Case structure [diff. entre positions brutes et filtre < 100mm] : $\overline{\overline{P_k}} = \overline{\overline{P_k}}$

Les images du VI du filtre de Kalman est disponible en Annexe 2.

Une certaine difficulté réside dans le fait de choisir les bons paramètres à savoir :

- À partir de quelle différence entre les 2 dernières positions brutes passer à la différence finie
- À partir de quelle différence entre la position brute et filtrée, réinitialiser le filtre
- La matrice d'estimation de la covariance de l'erreur R
- La matrice $\overline{\overline{P_{add}}}$

Une matrice R trop importante et une réinitialisation trop rapide, crée des « saut » dans la position et la vitesse (Figure 8). Ce qui est un peu contre-intuitif, la tendance, en présence d'imprécisions serait d'augmenter R.

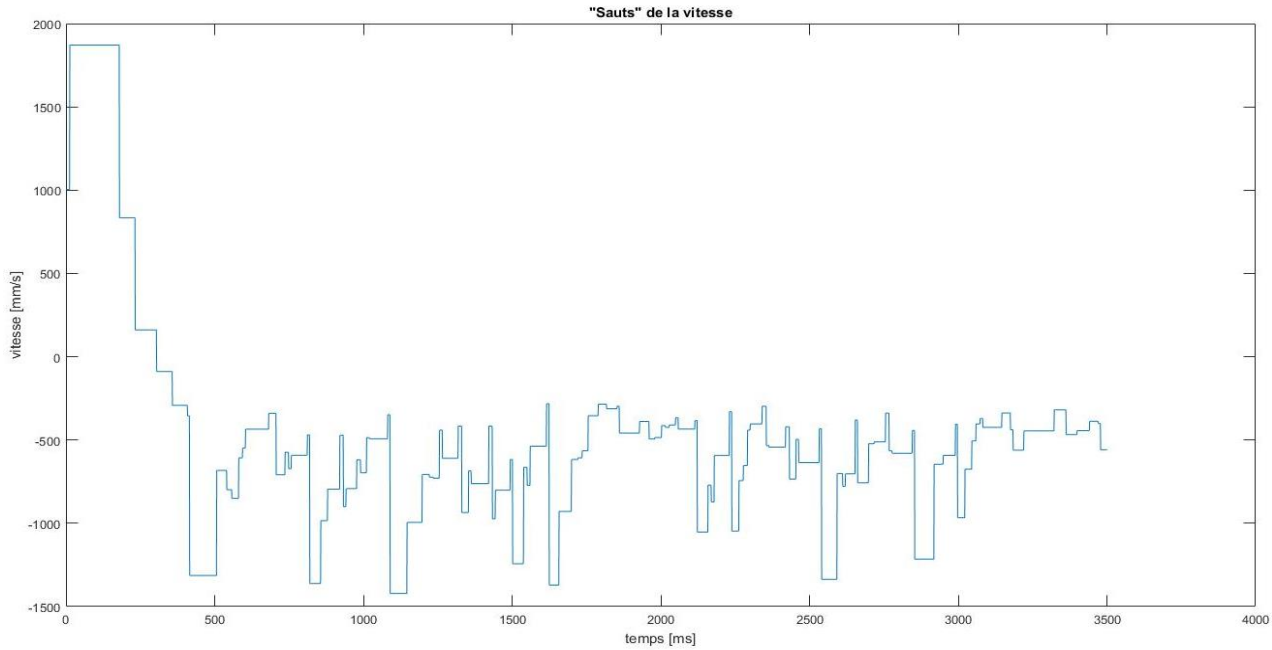


Figure 8 –Vitesse X, de magnitude moyenne

En diminuant R à $\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 8 & 0 \\ 0 & 0 & 0 & 8 \end{bmatrix}$ et en augmentant la distance de réinitialisation à 150mm, nous obtenons des résultats légèrement meilleurs (pour le filtrage de la vitesse, Figure 9), en tout cas pour des vitesses de 500 mm/s (2ème courbe de la Figure 9). Pour la première et troisième courbe, nous pouvons voir que la vitesse réagit vite mais de façon imprécise. Cela vient du fait que la différence finie a été appliquée (vu que la différence de position entre les 2 dernières valeurs est supérieure à 50mm), et de la fréquence faible d'actualisation.

La flèche indique un plateau dans lequel la position n'a pas été actualisée pendant 151ms, ce qui crée un fort saut de la vitesse lors de l'acquisition d'une nouvelle donnée.

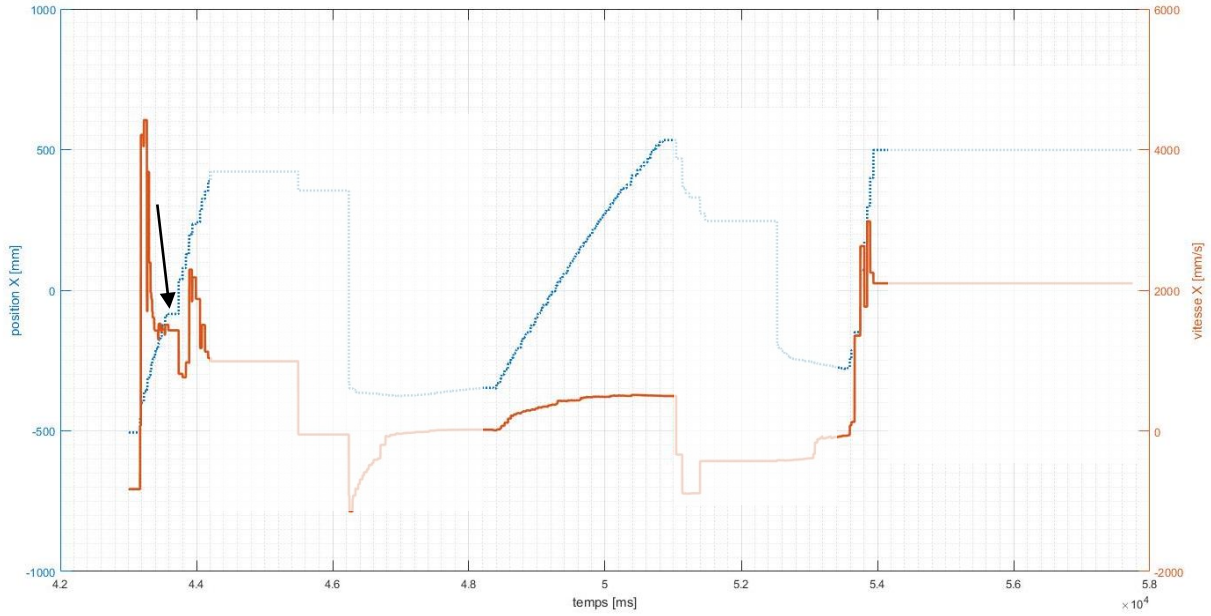


Figure 9 – 3 lancer de balle le long de l'axe X, avec la position brute et la vitesse filtrée (les traitilllées n'indiquent pas forcément des points de mesures)

Continuation

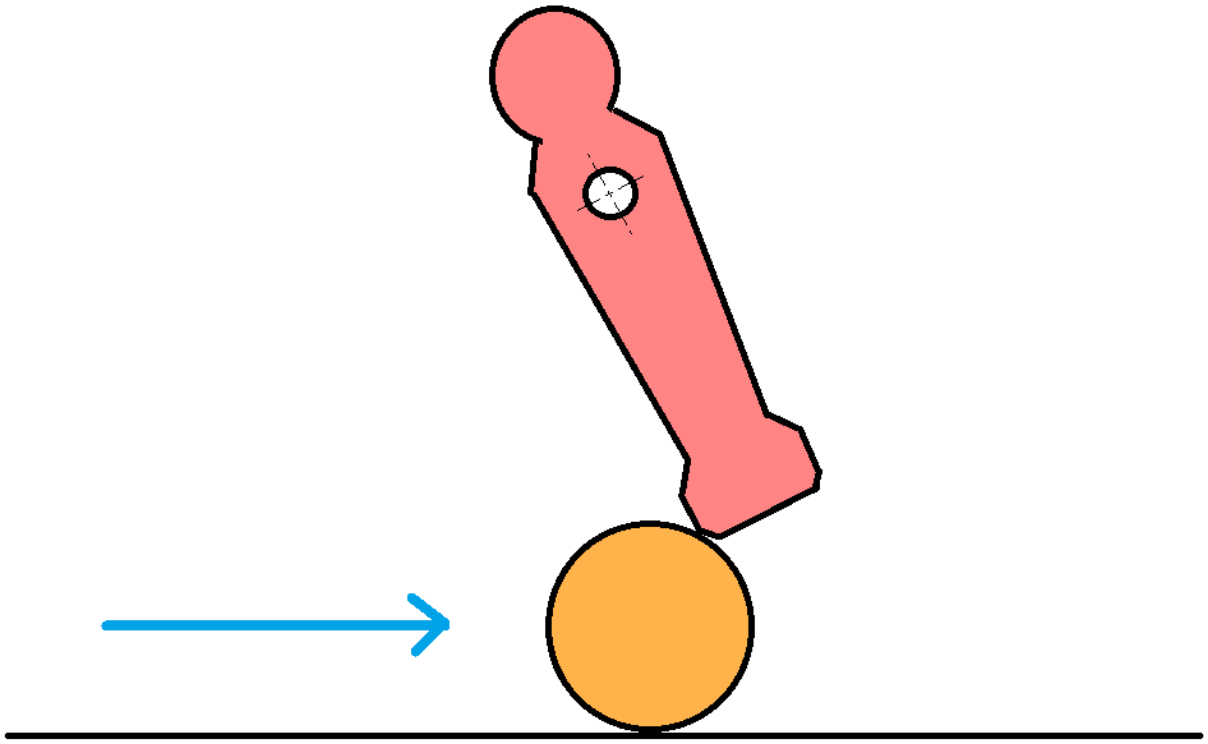
La création d'un filtre qui prédise la position et la vitesse de manière précise pour toutes les plages de vitesse, n'a pas complétement abouti. En particulier la différence finie non-filtrée n'est pas une solution pour acquérir la vitesse.

Les imprécisions proviennent en partie d'un choix non-optimal des constantes, mais également de l'imprécision des données brutes elles-mêmes (Chapitre sources d'erreurs). Avant de choisir des meilleures constantes, il est conseillé d'améliorer si possible la partie vision de la balle. Nous en parlerons dans le chapitre : prochains travaux.

Blocage de balle

Beaucoup de stratégies possibles passent par l'arrêt et le contrôle de la balle. Bien que par la suite nous n'ayons pas utilisé de stratégies de blocage, nous avons rajouté ce chapitre, dans la supposition qu'il pourrait être utile pour les prochaines personnes qui s'occupent de la stratégie.

Il a été observé qu'il est possible d'arrêter une balle en penchant le joueur bloqueur comme montré dans le schéma ci-dessous.



Nous avons fait une expérience afin de déterminer quel angle il faudrait imposer aux joueurs afin de bloquer une balle arrivant perpendiculairement à l'axe du joueur.

Procédure de l'expérience

L'angle testé a été imposé à l'aide du programme LabVIEW, sur le joueur de l'ordinateur que nous voulons tester.

Nous avons tiré avec un joueur humain se trouvant en face du joueur avec lequel on veut bloquer la balle.

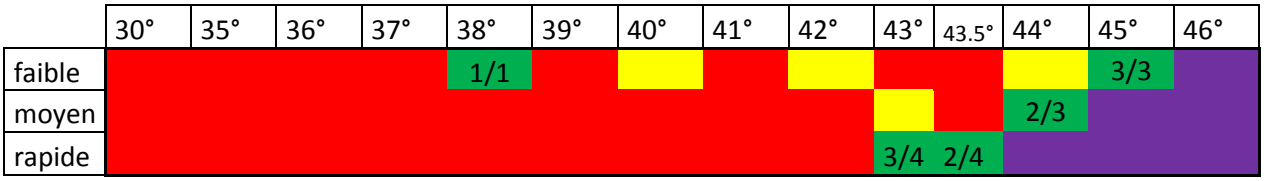
Plusieurs tirs ont été effectués pour chaque angle testé, avec des vitesses différentes.

Les tirs ont été divisés en 3 catégories, avec la vitesse à titre indicatif :

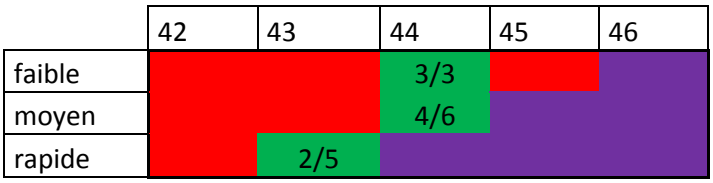
- Tirs forts (~ 5.5[m/s] à 2.5[m/s])
- Tirs moyens (~ 2[m/s] à 1[m/s])
- Tirs faibles (~ 0.5[m/s] à 0.1[m/s])

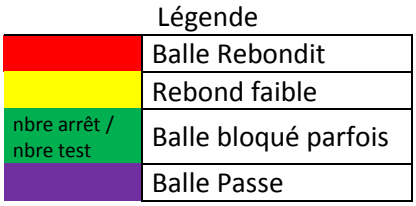
Résultats

Attaquants :



Milieux :





Analyse des résultats

Nous trouvons les angles optimaux suivants :

	Tirs forts	Tirs moyens	Tirs Faibles
Milieux	43°	44°	44°
Attaquants	43°	44°	45°

Bien que la méthode ne semble pas fonctionner à tous les cas, nous pouvons en tirer que l'angle de tir optimal se trouve dans les 44°, et que suivant la force du tir, il faudrait le régler de plus ou moins 1°. Nous pourrions l'expliquer par le fait que sous une certaine force, les barres fléchissent un peu. Et de la présence d'un léger jeu avec le roulement.

Des erreurs dans ces résultats peuvent être dus au fait que, même si l'angle mesuré par le programme LABVIEW correspond bien à celui imposé dans ce même programme, la référence peut bouger un peu au cours de l'expérience si l'axe est mal serré. En effet, ces résultats sont tirés de la seconde fois que nous avions réalisé l'expérience, la première fois les accouplements des axes se déformaient sous les chocs.

D'autres sources d'imprécisions proviennent du fait que la barre des joueurs est légèrement tordue en particulier celle des défenseurs, de plus les variations de forces du joueur humain sont à prendre en compte. La balle n'étant plus une sphère parfaite, elle ne possède pas le même diamètre à 0.1 mm près.

Renvoi de la balle

Introduction

Dans cette partie nous proposerons un modèle mathématique de notre système balle-joueur, et son fonctionnement pour renvoyer la balle dans une direction précise. Dans le sous chapitre « coordonnés du joueur lors du contact avec la balle », nous présenterons un modèle tridimensionnel du shoot de la balle, et nous le comparerons avec le modèle bidimensionnel. Puis à l'aide de la réponse dynamique des joueurs nous montrerons un VI de tir implémenté mais non fonctionnel.

Présentation du modèle

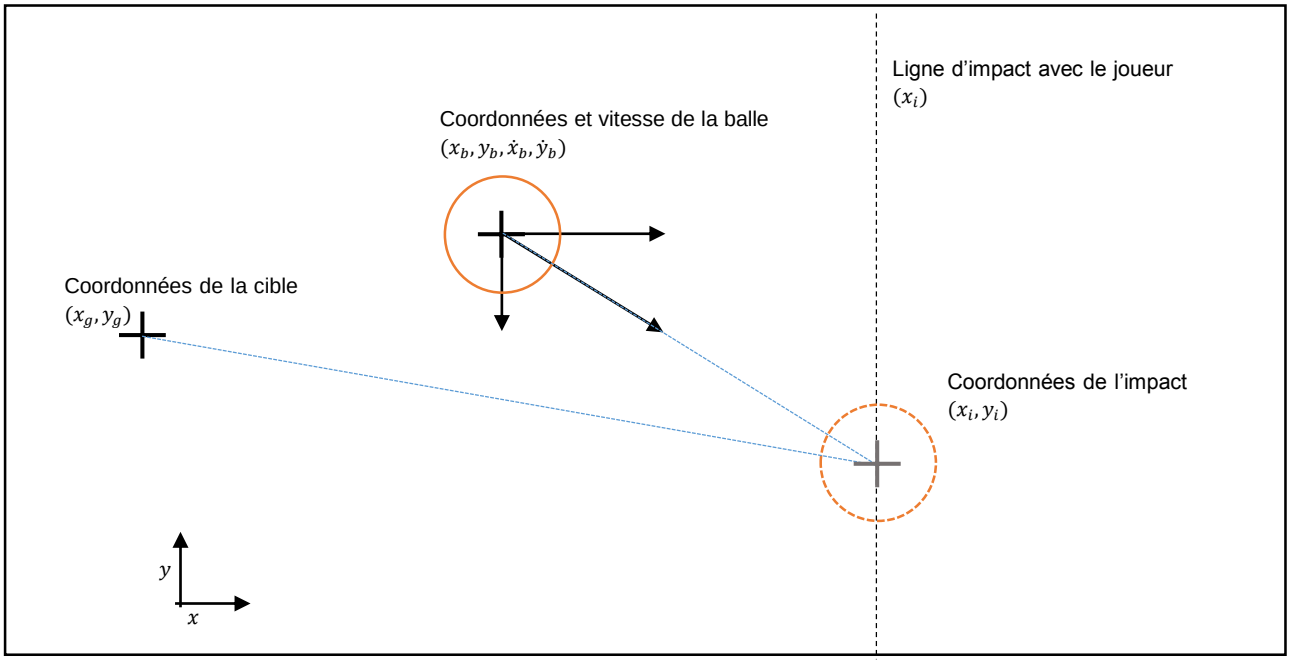


Figure 10

Comme présenté dans la Figure 10, nous connaissons trois entrées : Les coordonnées et la vitesse de la balle, les coordonnées de la cible et la ligne d'impact avec le joueur. Par souci de simplification, nous avons fait coïncider cette dernière avec la coordonnée en x de l'axe des joueurs.

À partir de ces informations il est possible de calculer le temps jusqu'à l'impact :

$$t_i = \frac{x_b - x_i}{\dot{x}_b}$$

Et la coordonnée transversale de l'impact : $y_i = y_b + \dot{y}_t * t_i$

L'étape suivante consiste à trouver les coordonnées du joueur pour pouvoir choisir sa position et sa dynamique pour pouvoir tirer au bon moment.

Étant donné que le pied seul du joueur touche la balle, nous pouvons modéliser le joueur comme un rectangle de 12x21 mm (Figure 1). Une hypothèse, à valider lors de test, est que l'inertie du pied est infinie, cela revient à dire que la vitesse et la direction de la balle avant l'impact n'importent pas sur la direction de la balle après l'impact.

Coordonnées du joueur lors du contact avec la balle

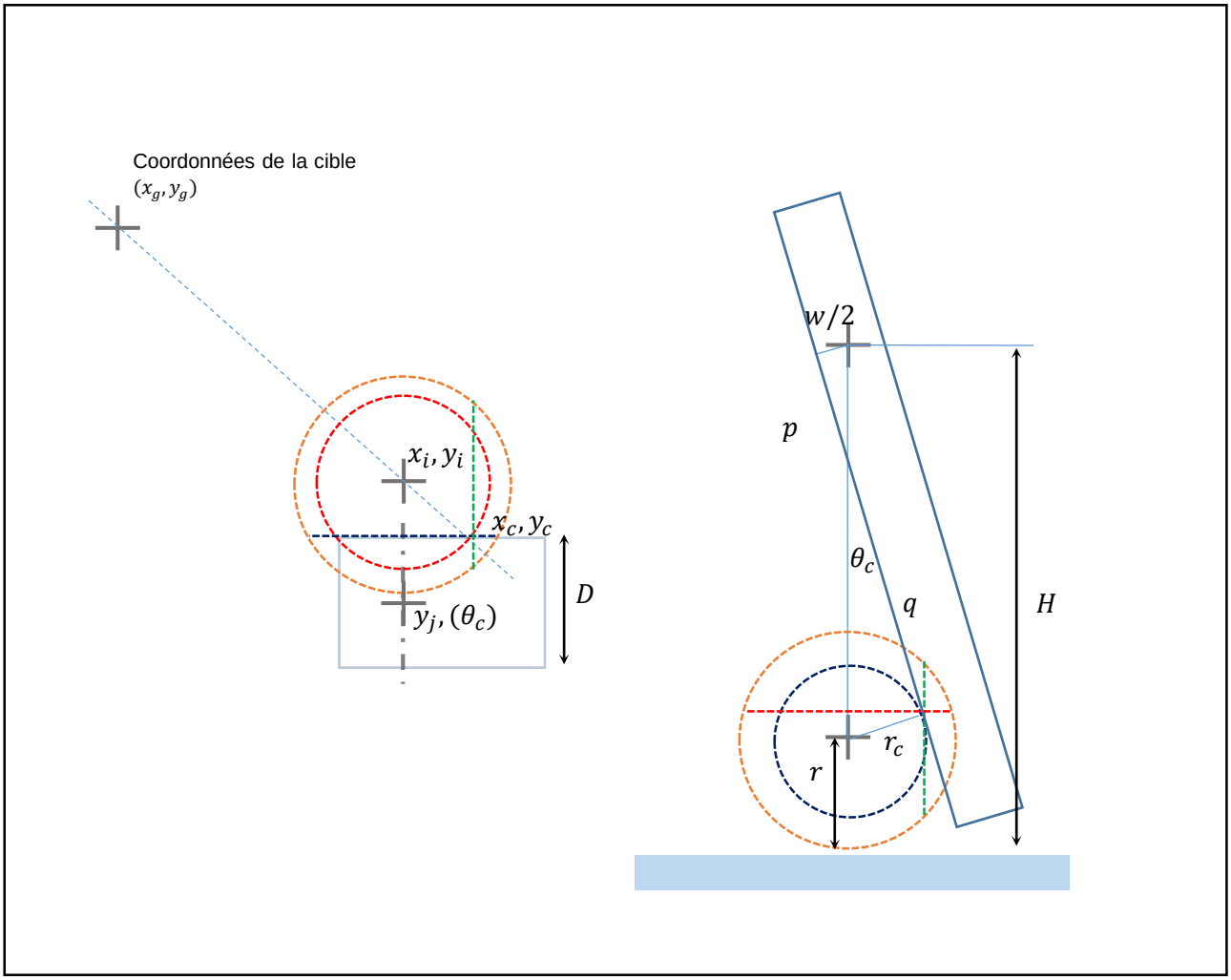


Figure 11 – Modélisation lors du contact de l'impact

Les équations que l'on peut tirer de la Figure 11 sont :

(1) $\frac{x_g - x_i}{y_g - y_i} = \frac{x_i - x_c}{y_i - y_c}$

(2) $x_i - x_c = r_c * \cos(\theta_c)$

(3) $\tan(\theta_c) = \frac{w}{2\,p} = \frac{r_c}{q}$

(4) $H = r + \cos(\theta_c) * (p + q)$

(5) $r^2 - r_c^2 = (y_i - y_c)^2$

But : trouver $\theta_c, y_c, (x_c, p, q, r_c)\ en\ fonction\ \ de\ x_g, y_g, x_i, y_i, w, H, r$

À partir de l'équation (3) nous pouvons définir $p = \frac{w}{2\tan(\theta_c)}\ et\ q = \frac{r_c}{\tan(\theta_c)}$ les insérant dans (4) nous obtenons :

(6) $H = r + \frac{\cos^2(\theta_c)}{\sin(\theta_c)} * (w/2 + r_c).$

Avec les équations (1), (2) et (5), il est possible de trouver l'équation suivante :

$$\frac{x_g - x_i}{y_g - y_i} = \frac{r_c * \cos(\theta_c)}{\sqrt{r^2 - r_c^2}}$$

Donc, que $\theta_c = \arccos\left(\frac{x_g - x_i}{y_g - y_i} * \frac{\sqrt{r^2 - r_c^2}}{r_c}\right)$. Insérons ce résultat dans (6) :

$$H = r + \frac{\left(\frac{x_g - x_i}{y_g - y_i} * \frac{\sqrt{r^2 - r_c^2}}{r_c}\right)^2}{\sin\left(arccos\left(\frac{x_g - x_i}{y_g - y_i} * \frac{\sqrt{r^2 - r_c^2}}{r_c}\right)\right)} * \left(\frac{w}{2} + r_c\right)$$

Il ne reste plus qu'à résoudre pour r_c. Malheureusement il n'existe pas de solution analytique. Pour cette raison, nous nous sommes tournés sur le programme Mathematiqua 10.3, (un fit avec Matlab aurait tout autant bien fonctionné).

Nommons les paramètres inconnus $\frac{x_g - x_i}{y_g - y_i} = a$, et l'angle de tir $\Omega = Arctan\left(\frac{1}{a}\right)$, plus intuitif que *a*.

Dès lors nous pouvons trouver numériquement r_c en fonction de Ω

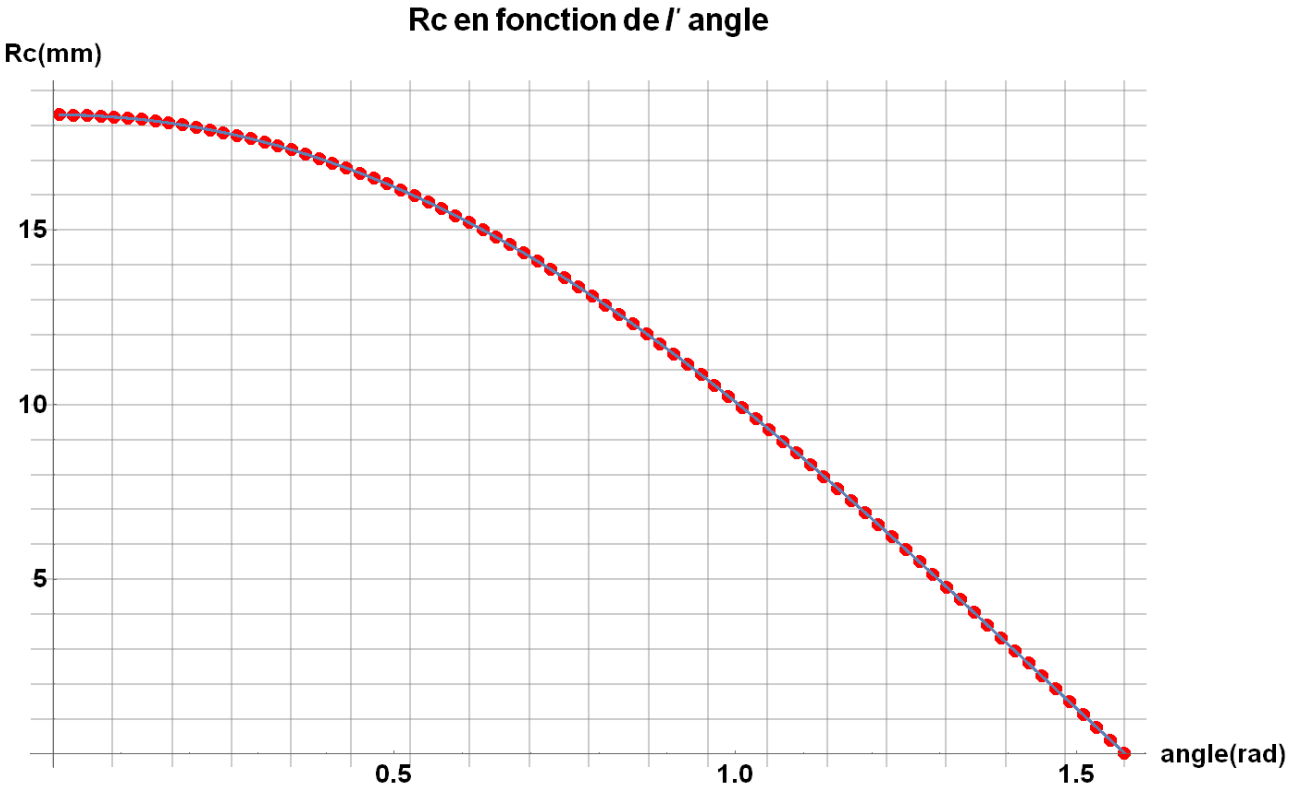


Figure 12- (r =18.3 mm, H=85 mm, D=12 mm)

Un polynôme de degré 4 retrace bien ces points (Figure 12). Ses coefficients ont les valeurs suivantes :

(7) $r_c(\Omega) = 18.3 - 0.08001 * \Omega - 7.428 * \Omega^2 - 2.036 * \Omega^3 + 1.324 * \Omega^4$

La question se pose si ce modèle complexe peut être simplifié par un modèle bidimensionnel. Et effectivement, si l'on compare les courbes de l'angle en fonction de $\Delta y = y_i - y_g$, de ce modèle et

du modèle ou l'angle de tir est tout simplement $\Omega = \arcsin\left(\frac{\Delta y}{r}\right)$, nous trouvons une erreur maximale de 1.16°, dans les environs de 10 mm montré dans la Figure 13.

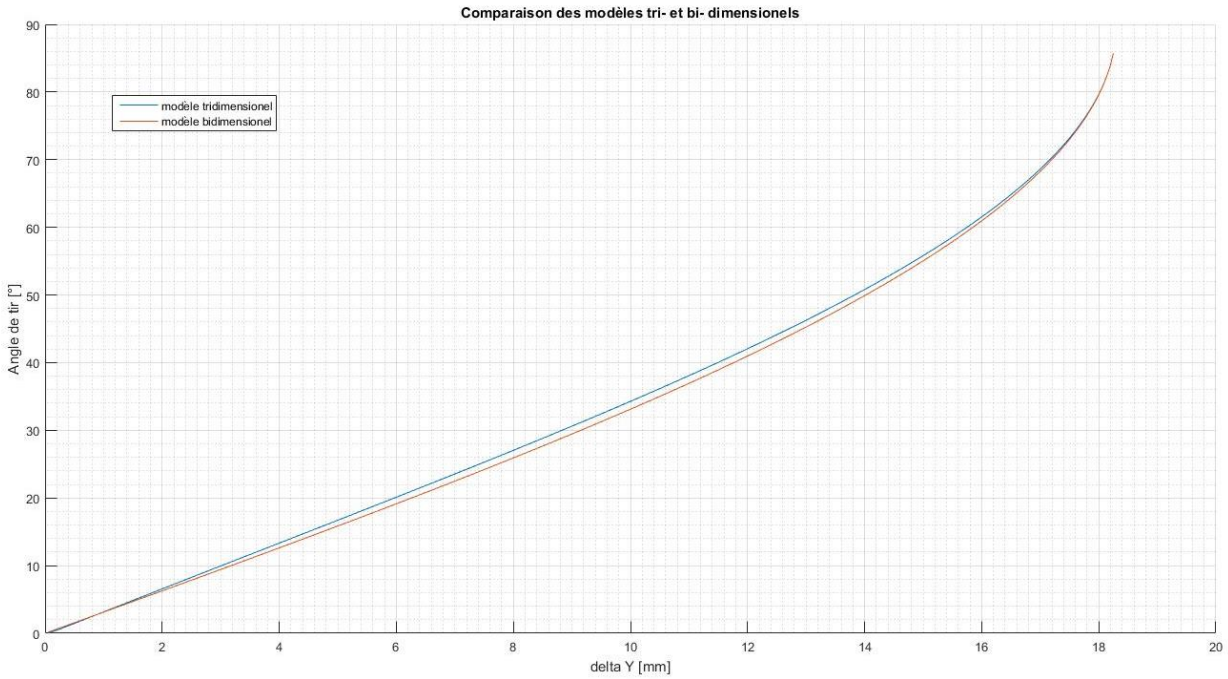


Figure 13

Le rapport « Fast Positionning ; MÉTRAILLER Simon, automne 2014 », allait dans cette direction, mais nous voulions nous assurer de la validation du modèle pour des angles supérieurs à 15°.

Le modèle bidimensionnel étant précis et beaucoup plus simple, nous choisissons dès lors de l'adopter.

L'angle r_c devient alors tout simplement :

(7') $r_c = r * \cos(\Omega)$

Ce qui nous permet finalement de calculer θ_c *et* x_j:

$$(8)\ \theta_c = \arcsin\left(\frac{r_c}{H - r}\right)$$

$$(9)\ y_j = y_c + \frac{D}{2} = y_i + r\sqrt{1 - \cos^2(\Omega)} + D/2$$

Dynamique du joueur

Pour finaliser le VI, Il s'agit de connaître la dynamique du joueur. À l'aide de ces données, nous pouvons savoir, à partir de quel moment il faut enclencher l'ordre de tir. Pour cela, nous avons sorti les valeurs temporelles de l'angle des joueurs dans un fichier texte (Annexe 4, dynamique d'autres joueurs), et comme précédemment nous créons une fonction qui approche aux mieux ces points.

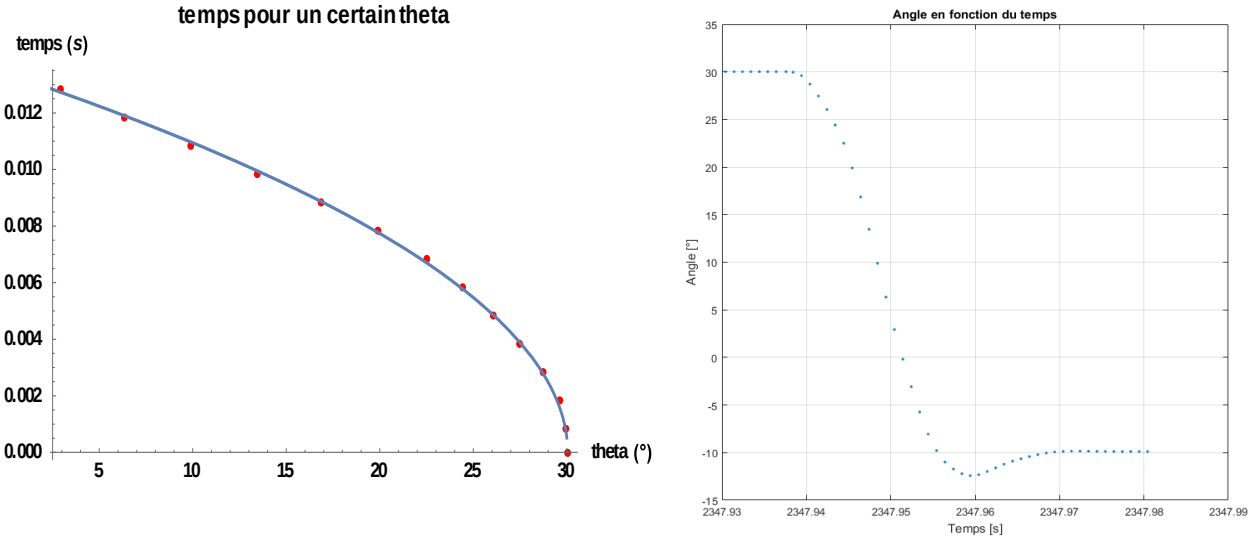


Figure 14 – Défenseurs, pour un changement de position de 30° à -10° et un PID de (0.5, 0.01, 0.001)

La formule approchée pour les défenseurs (Figure 14, ligne bleue) est la suivante :

$$t[s, grande\ précision] = -0.000033856 + 0.0024515\sqrt{30.05 - \theta}$$

$$t[ms, précision\ 0.1ms] = 2.5\sqrt{30.05 - \theta}$$

Nous pouvons dès lors, trouver le temps du tir ($t_t(\theta = \theta_c)$).

Récapitulation

Entrées :

- Position et vitesse de la balle
- Cible à toucher
- Coordonné x, des joueurs concernées

Calculs :

- Temps jusqu'à l'impact (t_i)
- Angle et position du joueur lors de l'impact
- Temps qu'il faut pour le joueur pour partir de sa position de départ jusqu'à l'angle de contact (t_t)

Sortie :

- Angle et position du joueur qui tire
- Ordre de tir déclenché à ($t_i < t_t$)

Le VI crée se trouve en Annexe 3.

Précision des données et sources d’erreurs

Introduction

Le VI de tir implémenté sur les défenseurs ne touchant pratiquement jamais la balle, et le positionnement du gardien dans la direction de la balle ne fonctionnant pas toujours, c’est pourquoi nous avons jugé bon de rajouter un chapitre sur des sources d’imprécisions.

Sensibilité de l’angle de tir en fonction de l’erreur entre joueur et balle

Il serait intéressant d’observer à quel point les données seraient influencées par une imprécision du paramètre y_j, pour plusieurs raisons :

- Le filtre a mal calculé positions et vitesses
- La précision de la caméra et de l’algorithme de détection de la balle.
- La précision des moteurs linéaires (point non vérifié)

Nous supposons la position de la balle sur l’axe des joueurs et nous introduisons une erreur relative (de la position y) entre la balle et le joueur. (Dans la direction externe de la balle, dans l’autre sens on tire tout droit).

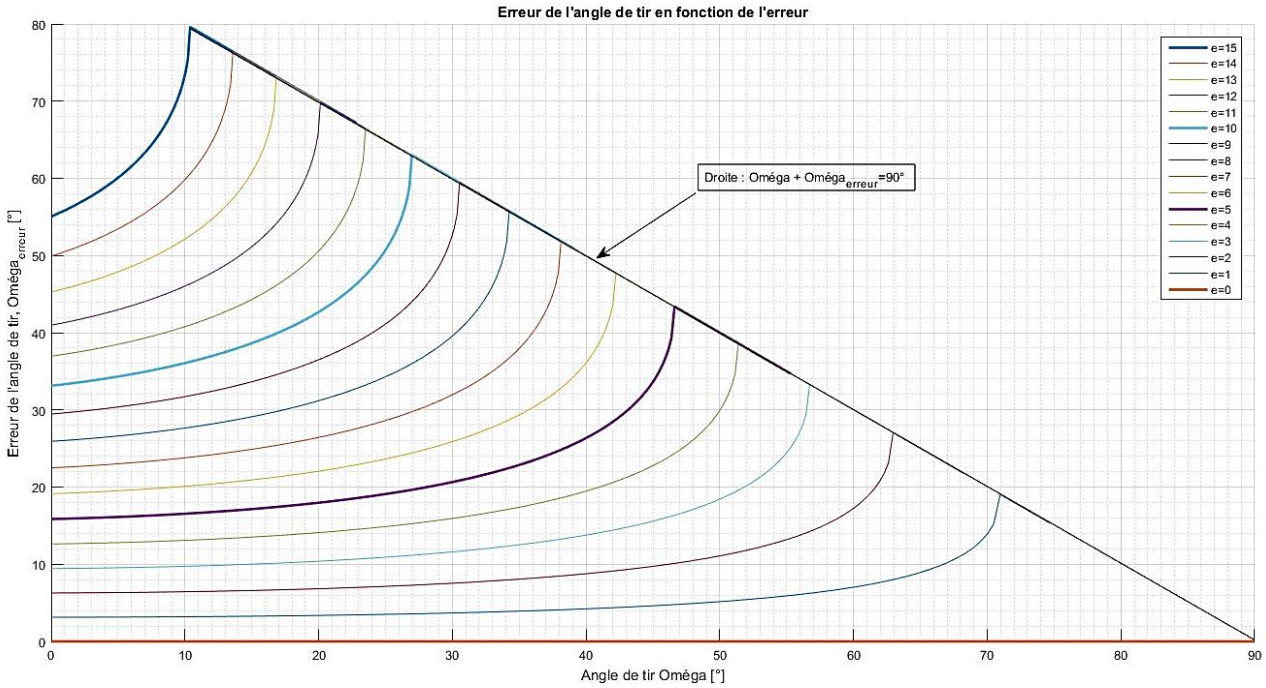


Figure 15

Nous pouvons donc voir à partir de la Figure 15, que l’erreur sur l’angle de tir augmente avec l’erreur de positionnement et l’angle de tir. Nous pouvons observer par exemple, qu’une erreur de positionnement de 4mm, génère une variation de 13° pour les faibles angles de tir.

Sources d’erreurs

Précision du VI de tir

Le temps de tir entre la position au repos et le contact avec la balle prend environ 15 ms. L’algorithme de traçage donnant la position entre 4 et 150 ms plus tard, il pourrait y avoir un retard important de l’ordre de tir si la vitesse de la balle est grande, d’autant plus si la vitesse est grande.

Précision de filtre

Pour filtrer un maximum la vitesse, nous avons choisi d’augmenter la distance de réinitialisation du filtre. Cela veut dire que la position de la balle pourrait avoir une différence de 100 mm avec la position réelle avant que le filtre ne réagisse, bien que la matrice R ait plusieurs fois été diminuée. Nous conseillons aux prochains utilisateurs d’ajuster ces deux derniers paramètres de telle sorte à ce que la position de la balle filtrée ne dévie pas plus de 5 mm avec la position brute.

Précision de la position brute

Dans le doute, nous avons vérifié la précision de la position de la balle brute. Pour ce faire nous avons positionné la balle sous l’axe des joueurs à une distance de 240 mm du bord. À partir de ces huit mesures, nous avons réalisé que plus nous nous approchions du bord du babyfoot, plus l’erreur de positionnement augmentait.

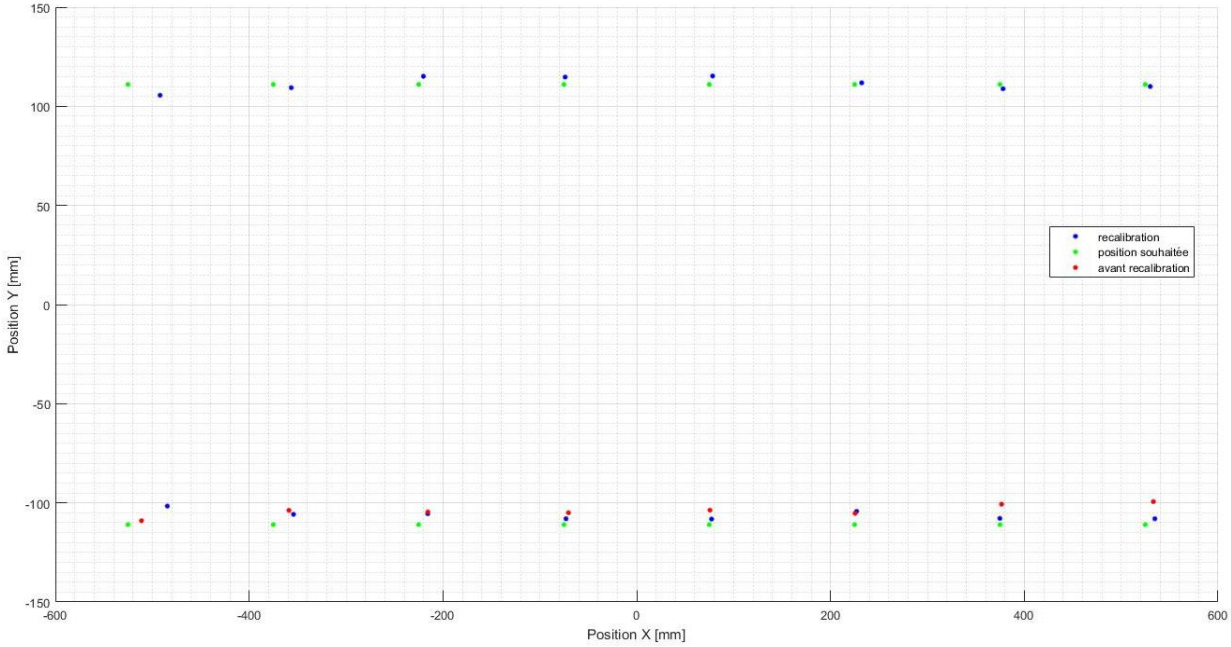


Figure 16 - Erreur de positionnement du système caméra/calibration

Nous avons choisi de refaire un calibrage, selon la procédure du document Babyfoot_upgrade_report.docx[2]. Et nous avons fait cette fois-ci seize mesures de part et d’autre du Babyfoot à 240 mm du bord. Ces valeurs avec celles avant calibrage sont regroupées en Annexe 5.

[2] “Upgrade of the babyfoot vision control system”, par Léo Sibut sous la supervision du Dr. Christophe Salzmann

Les différents points sont montrés dans la Figure 16. Nous pouvons donc calculer la précision de notre système :

Erreur de positionnement moyenne		//	variance
Avant calibration :	11.1 mm	//	136.9 mm
Après calibration :	11.6 mm	//	258.6 mm

La recalibration n'a pas eu l'effet escompté, il est même légèrement de qualité inférieure. L'imprécision la plus forte étant au niveau du gardien (~14 mm avec l'ancienne calibration). Cela pourrait expliquer une des raisons pour laquelle le gardien a des difficultés de se positionner au bon endroit.

Nous avons pu observer que la calibration de la couleur ou du taux de saturations doit être vérifiée dans les coins du babyfoot. Les reflets sur les barres des joueurs provoquent une déformation de l'image filtrée par saturation de la balle.

Conclusion

Au terme de ce projet nous pouvons exprimer le bilan suivant :

- L'arrêt de la force après 35 ms de blocage fonctionne bien.
- Le filtre est effectif pour des vitesses faibles à moyennes. Pour des vitesses trop grandes la vitesse devient discontinue et cela pour plusieurs raisons :
    - Choix sous-optimal des constantes. Il est difficile des choisir les bons paramètres car il faut faire de nombreux compromis, et jongler entre 4 constantes.
    - Fréquence d'acquisition de la balle trop faible pour l'algorithme de « pattern matching ». Une fréquence d'actualisation qui descend jusqu'à 6.6 Hz donne trop peu de points pour que le filtre fonctionne correctement.
- La balle pourrait être arrêtée par blocage, si l'on inclut un certain pourcentage d'échec. Mais dans ce cas il faudrait savoir comment la contrôler.
- La différence de l'angle de tir pour la balle sous l'axe des joueurs - entre les modèles bidimensionnel et tridimensionnel - est négligeable
- Si l'usage d'un algorithme devant tirer la balle avec un angle se voulait à être implémenté, il faudra porter une grande sensibilité à diminuer toutes les sources d'erreur, qui mêmes petites, ont un grand impact sur la variation d'angle.
- Les valeurs de la position de la balle sont imprécises en moyennes de 11mm. Avec les positions les plus imprécises aux bords du terrain.

Bien que les VI crées n'ont eu que peu d'impact sur la qualité de jeu de l'ordinateur, nous espérons qu'ils seront utiles pour les prochaines personnes s'occupant du babyfoot.

Il a été très intéressant de travailler dans un VI autant complet, et nous sommes reconnaissants à toutes les personnes qui ont construit, amélioré, supervisé, réparé, et aidé à la réalisation de ce projet.

Prochains travaux / notes pour les prochains étudiants

Nous avons listé ci-dessous des points que nous avons jugés importants :

- L'ajout d'un couvercle semi-réfléchissant ou en plexiglas teintée, la fermeture du bas du babyfoot, et l'ajout d'une source diffuse de lumière depuis le bas (néons ou autre source de lumière avec diffuseur), pourraient drastiquement diminuer la sensibilité de la caméra au bruit de la luminosité extérieure. Cela pourrait également diminuer la variation de l'éclairage de la balle qui risque d'être perdue proche des gardiens. Cela permettrait également une acquisition de la position plus rapide, une précision accrue de la position de la balle, et un algorithme de tracking plus efficace.
- Lors de la calibration de l'image de référence, une source de lumière diffuse devrait être présente pour avoir le meilleur résultat possible.
- Lorsque la position de la balle sera connue de manière plus sûre, il sera peut-être possible de diminuer la matrice de l'estimation de la covariance de l'erreur R, tout comme la distance de réinitialisation du filtre, pour de meilleurs résultats.
- La partie du filtre qui actualise la vitesse - si la différence de position est trop grande -, devrait être superflue, et une source d'imprécision de la vitesse pourrait être éliminée.
- Il serait possible de mettre le filtre de Kalman directement dans la « Vision loop » pour actualiser les données exactement lors de nouvelles mesures, cela permettrait également de faire une discrétisation de la surface du jeu, comme proposé par le rapport « Vision de la balle », qui augmenterait la vitesse de prédiction.
- Des cours de LabVIEW gratuits sont organisés pour les étudiants. Ils sont d'une grande aide lorsque les notions de programmation de première année remontent un peu.
- Les points du rapport « vision de la balle » sont pertinents, et leur analyse vaut la peine d'être considérée.
- Écrire un VI sur LabVIEW 2015 et l'implémenter sur le projet de babyfoot qui tourne avec la version LabVIEW 2014 ne fonctionne pas.
- Il est possible de créer des fichiers textes avec les valeurs d'intérêt. (Deux de ces façons sont décrites dans l'Annexe 6)

ANNEXES

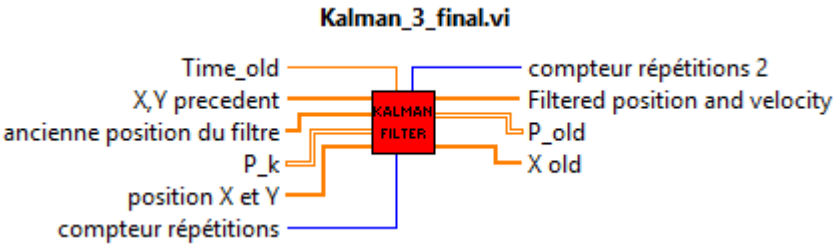
Annexe 1 – Autres grandeurs du Babyfoot

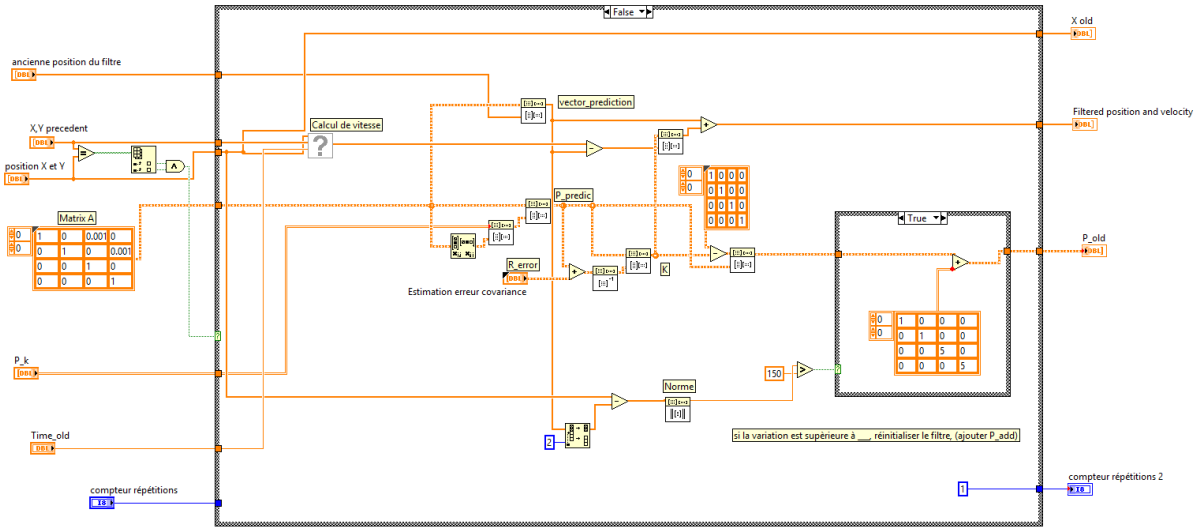
Déplacement latéral maximal des joueurs (vérifié) et distance entre joueurs (non vérifié, pris des VI existants) :

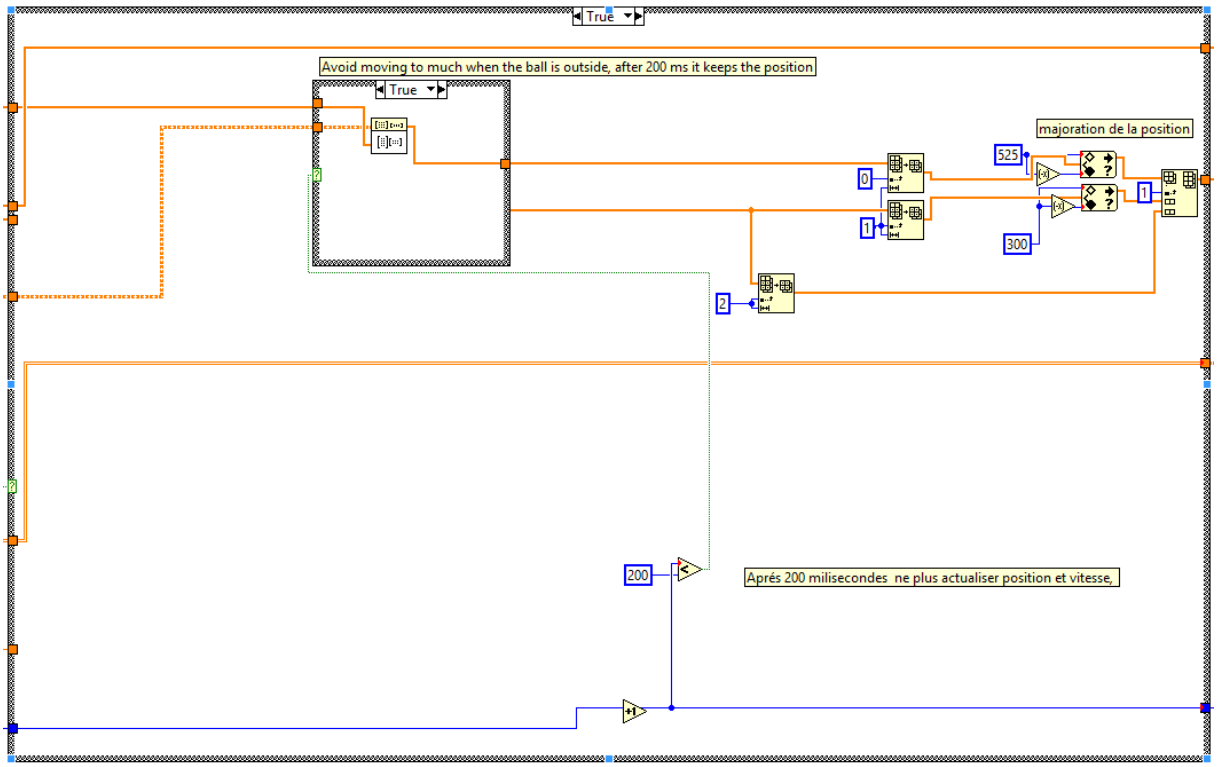
	Déplacement maximal	Distance entre joueurs
Gardien :	100 mm	-
Défenseurs :	170 mm	2*119 mm
Milieux :	65 mm	119 mm
Attaquants :	94 mm	207 mm

Rayon de la balle : 18.3 mm

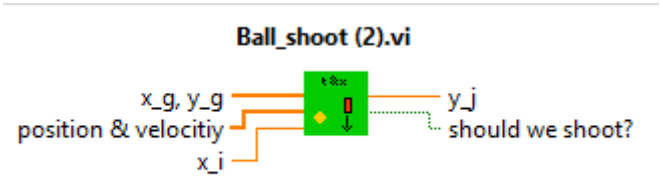
Annexe 2 – VI du filtre de Kalman

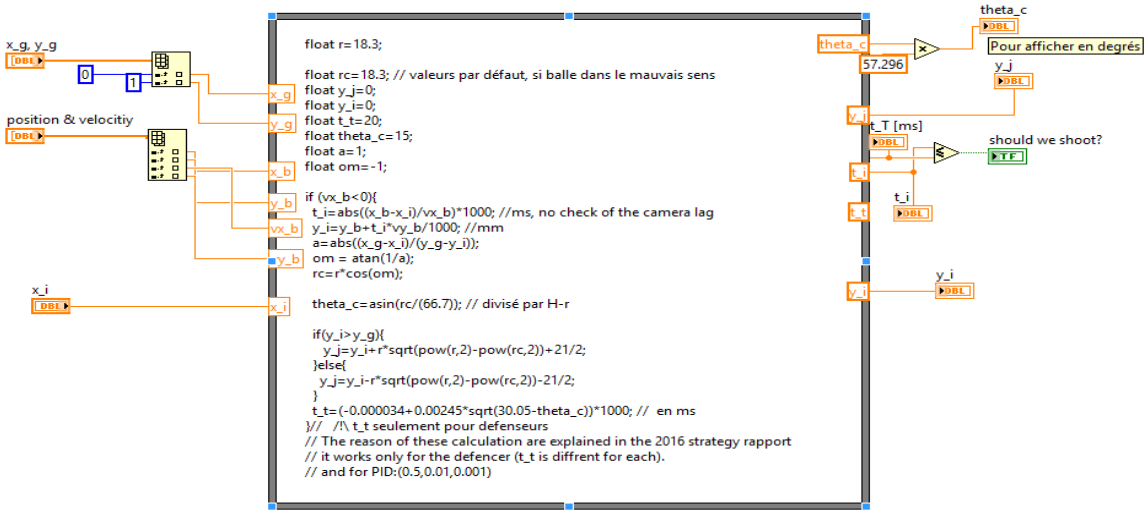




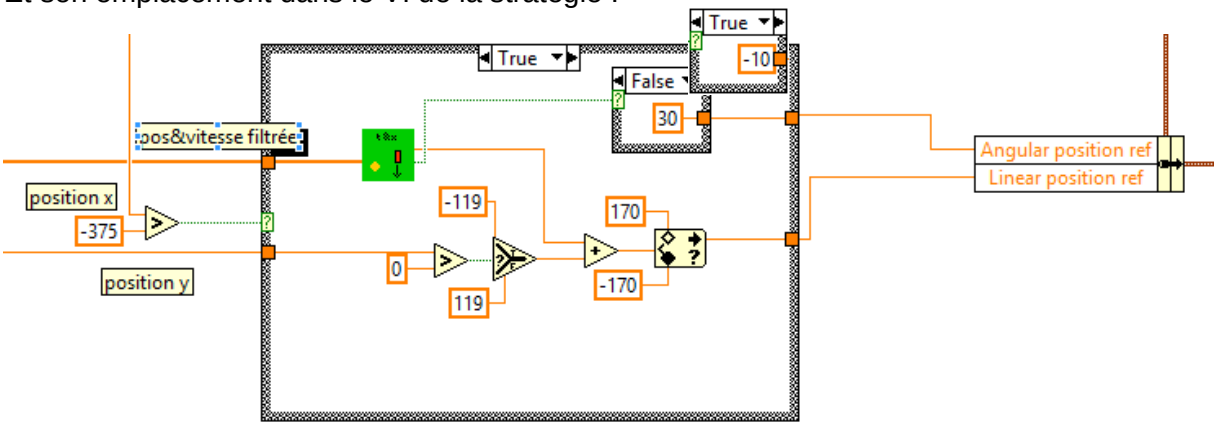


Annexe 3 – VI du tir direct de la balle

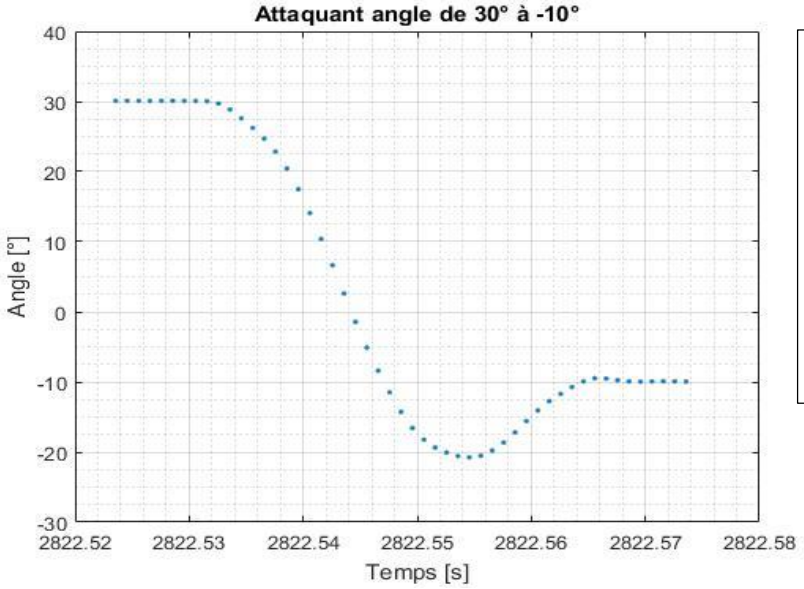




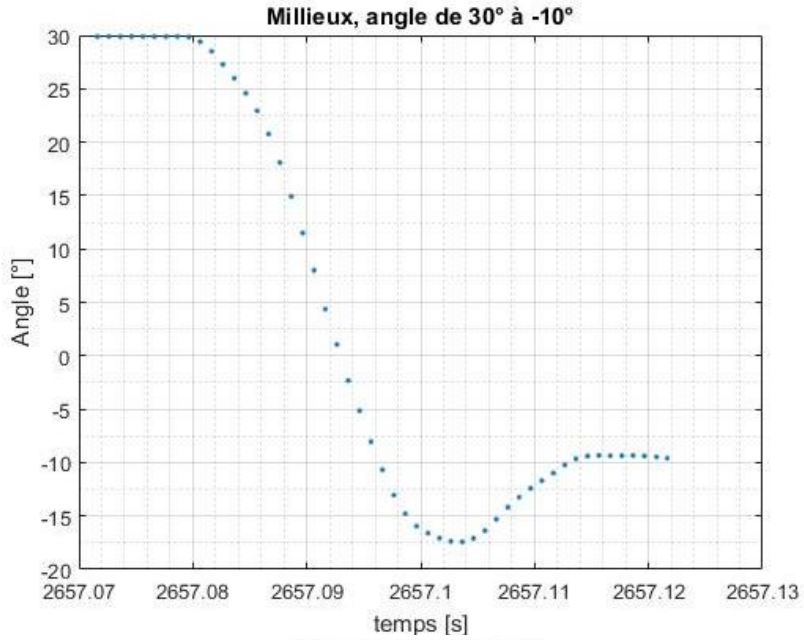
Et son emplacement dans le VI de la stratégie :

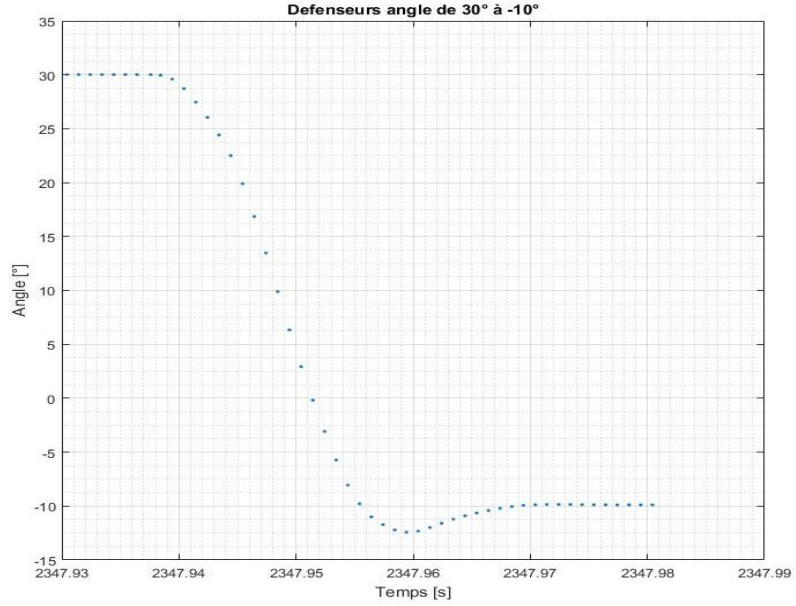


Annexe 4 – Variations d'angles pour un PID de (0.5, 0.01, 0.001)



D'autres angles ont été testés. (40° à 0°, 40°à-10°,30° à 0°, 20°à 0° et 20°à -10°). Et sont disponibles dans un dossier Babyfoot_2016-Copy/W11_theta_time, avec les données brutes. La commande était majorée entre -10 et 10.



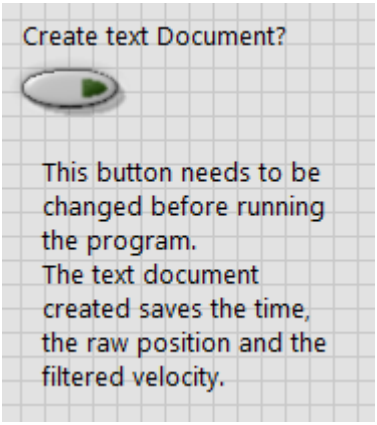


Annexe 5 – comparaisons entre vraies positions et positions avant et après calibrage

Position réel	X	-525.0	-375.0	-225.0	-75.0	75.0	225.0	375.0	525.0	525.0	375.0	225.0	75.0	-75.0	-225.0	-375.0	-525.0	
	Y	-111.0	-111.0	-111.0	-111.0	-111.0	-111.0	-111.0	-111.0	111.0	111.0	111.0	111.0	111.0	111.0	111.0	111.0	
Avant calibrage	X	-511.1	-358.9	-215.5	-70.4	75.6	225.4	376.6	533.3									Moyenne	Variance
	Y	-109.0	-103.8	-104.6	-105.0	-103.7	-105.3	-100.7	-99.4										
	Erreur	14.0	17.6	11.5	7.6	7.3	5.7	10.4	14.3									11.1	136.90
Après calibrage	X	-484.4	-354.3	-215.7	-73.0	77.3	227.0	374.9	534.8	530.0	378.1	232.2	78.4	-73.9	-220.1	-356.5	-492.0	Moyenne	Variance
	Y	-101.6	-105.8	-105.5	-108.0	-108.2	-104.3	-107.9	-108.0	110.0	108.8	111.8	115.3	114.7	115.2	109.4	105.6		
	Erreur	41.6	21.4	10.8	3.6	3.6	7.0	3.1	10.2	5.1	3.8	7.3	5.5	3.9	6.4	18.5	33.4	11.6	258.6

Annexe 6 – VI sortant un fichier texte

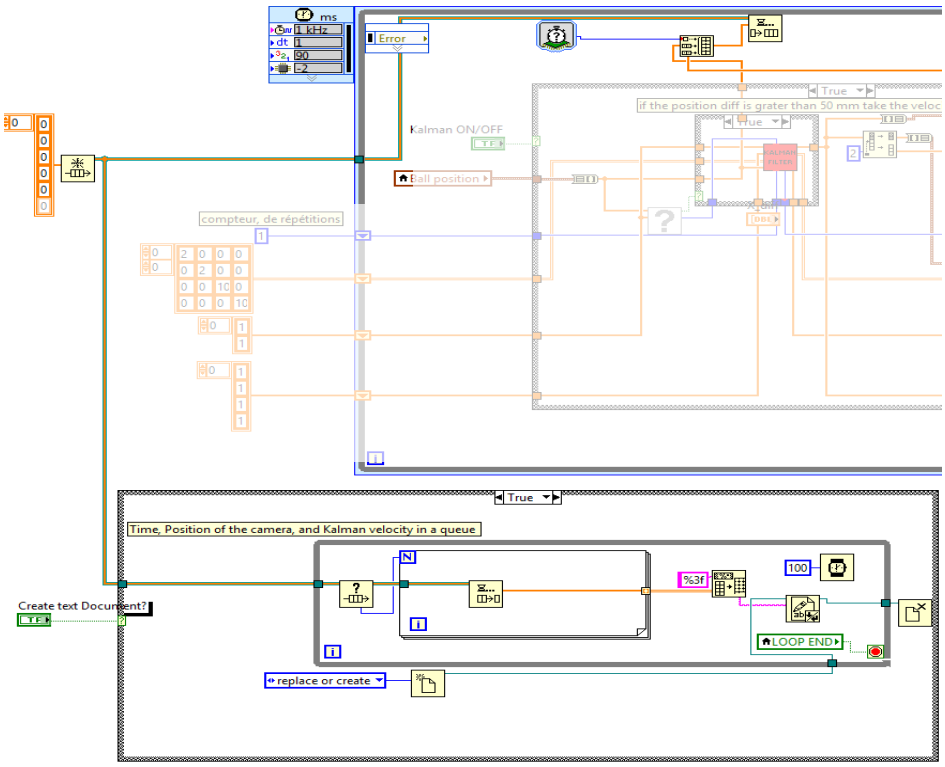
Deux possibilités pour créer un fichier texte à partir des données.



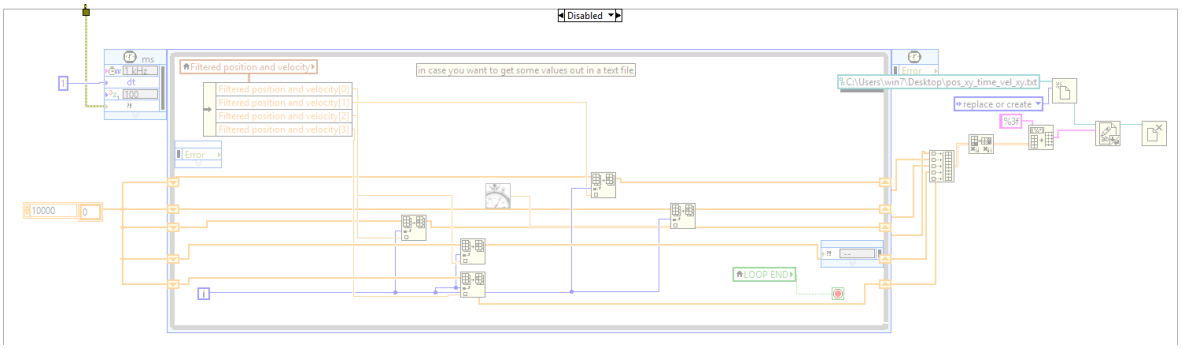
La première façon, consiste à presser le bouton « Create text Document ? » du Front Panel avant de démarrer le programme. Ce dernier créera une queue avec dans l'ordre d'apparition :

- Le temps
- La position X de la caméra
- La position Y de la caméra
- La vitesse X du filtre
- La vitesse Y du filtre

Une fenêtre pop-up s'ouvrira, et il est possible de choisir l'emplacement du fichier créé et son nom (par exemple : test12_R_petit.txt)



La seconde façon est de désactiver la « disable structure » créée autour de la boucle d'acquisition tout en bas du programme et de sortir via des variables globales les valeurs désirées.



Addendum

Avec une série de test, la précision du filtre a été améliorée et le VI de tir est devenu fonctionnel.

Les paramètres choisis sont les suivants :

- $R = \begin{bmatrix} 0.1 & 0 & 0 & 0 \\ 0 & 0.1 & 0 & 0 \\ 0 & 0 & 7 & 0 \\ 0 & 0 & 0 & 7 \end{bmatrix}$
- Distance de réinitialisation : 60 mm
- Différence pour le passage à la différence finie : 70 mm
- $P_{add} = \begin{bmatrix} 0.2 & 0 & 0 & 0 \\ 0 & 0.2 & 0 & 0 \\ 0 & 0 & 2 & 0 \\ 0 & 0 & 0 & 2 \end{bmatrix}$

Ces paramètres privilégient une vitesse stable par rapport à une actualisation rapide de la vitesse. (L'utilisateur favorisant le contraire pourrait diminuer l'estimation de la covariance de l'erreur sur la vitesse R[3,3] et R[4,4] et augmenter les valeurs Padd[3,3] et Padd[4,4].)

Les VI's de tir et de stratégies ont également reçu les modifications suivantes :

- Faire réagir les joueurs uniquement si le temps d'impact dure moins de 0.4 secondes
- Pour sélectionner le joueur, prendre le y_j à la place de y_b
- Lever les joueurs à partir de -385 mm à la place de -375 mm (une certaine marge puisque l'erreur moyenne de la balle est dans les 11 mm)

La précision de tir n'est par contre pas élevée, suivant la qualité du choix de la saturation, la balle va dans la bonne direction ($\pm$ 10°) dans la moitié ou le tiers des cas.

Cette précision pourrait être augmentée, comme mentionné précédemment, avec une meilleure calibration de l'image, voire une meilleure caméra.

Figure 17

L'usage d'un néon et de papier a également permis de tester une distribution plus régulière de la lumière (Figure 17). Cela permet une forme d'acquisition plus régulière de la balle dans les bords du babyfoot.

Nous conseillons aux prochains utilisateurs d'installer un dispositif similaire, et non provisoire